VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Možnosti využití vývojové platformy Nios II pro výuku a aplikace v biomedicínské technice

Possibilities of Using Nios II Development Platform for Teaching and Applications in Biomedical Engineeringe

2018 **Patrik Ponikva**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Patrik Ponikva**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Možnosti využití vývojové platformy Nios II pro výuku a aplikace v biomedicínské technice
Possibilities of Using Nios II Development Platform for Teaching and Applications in Biomedical Engineering

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Seznámení se s technikou FPGA a návrhem programovatelných logických obvodů.
2. Vypracování rešerše s tématem srovnání platforem programovatelných logických obvodů výrobců Xilinx a Altera.
3. Rozbor možností logiky FPGA a periferních obvodů vývojového kitu Nios II pro použití v biomedicínské přístrojové technice.
4. Návrh demonstrační úlohy s barevným maticovým LCD displejem.
5. Implementace demonstrační úlohy do vývojového kitu Nios II.
6. Oživení a experimentální ověření navržené úlohy.
7. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] ALTERA. *Embedded Design Handbook* [online] ED_HANDBOOK 2017.06.12 [cit. 2017- 06-30]. Dostupné z: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/edh_ed_handbook.pdf.
[2] ALTERA. *Nios II Embedded Evaluation Kit, Cyclone III Edition: User Guide* [online]. San Jose, July 2010 [cit. 2017- 06-30]. Dostupné z: http://www.altera.com/literature/ug/niosii_eval_user_guide.pdf.
[3] ALTERA. *Cyclone III FPGAs* [online]. [cit. 2017- 06-30]. Dostupné z: https://www.altera.com/products/fpga/cyclone-series/cyclone-iii/features.html.
[4] ALTERA. *Cyclone III Device Handbook: Volume 2* [online]. San Jose, July 2012 [cit. 2017- 06-30]. Dostupné z: http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf.
[5] KAŠÍK, Vladimír. *Programování hradlových polí.* Učební text a návody do cvičení. Ostrava: VŠB-TU Ostrava, 2012.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Vladimír Kašík, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Poděkování

Rád bych poděkoval panu Ing. Vladimíru Kašíkovi, Ph. D. za odbornou pomoc a konzultace během realizace této bakalářské práce.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Datum: 30. 4. 2018

Podpis studenta: 

Abstrakt

Tato bakalářská práce se zabývá možnostmi využití vývojového kitu Nios II řady Cyclone III od firmy Altera v biomedicíně a srovnáním platforem výrobců Altera a Xilinx. Stručně popisuje také základní strukturu FPGA čipů a poukazuje na konkrétní biomedicínské příklady, kde je možné tento vývojový kit využít. Praktická část se zabývá návrhem a realizací demonstrační úlohy, která využívá maticový LCD displej umístěný na multimediální desce kitu. Úkolem demonstrační úlohy je vytvoření ukázkového zobrazení pro monitor životních funkcí. Logika navržená v hradlovém poli je přizpůsobena pro využívání veškerých periferií na základní desce kitu a také samotného LCD.

Abstract

This thesis consist of options of the usage development kit Nios II Cyclone III series made by Altera for biomedical purposes and compares platforms made by Altera and Xilinx. Briefly describes basic structure of FPGA chips and shows specific biomedical examples of where is the usage of this development kit possible. Practical part is composed of desing and final realisation of demo task, which uses matrix LCD display placed on multimedial board kit. Purpose of this task is maintance of simulation for vital functions display. Logic designed in field programmable gate array is adapted for avail of all peripherals on motherboard of the kit and LCD.

Klíčová slova

FPGA, hradlová pole, biomedicína, Altera, Xilinx, Nios II, NEEK, Cyclone III, LCD.

Key words

FPGA, gate array, biomedicine, Altera, Xilinx, Nios II, NEEK, Cyclone III, LCD.

Obsah

Seznam použitých symbolů a zkratek **9**

Seznam obrázků **10**

Seznam tabulek **10**

Seznam grafů **10**

1 Úvod **11**

2 Technika FPGA a návrh programovatelných logických obvodů **12**
2.1 Kombinační logika . . . 12
2.2 Logické bloky . . . 13
2.3 Realizace paměti RAM . . . 14
2.3.1 Bloková RAM paměť . . . 14
2.3.2 Distribuovaná RAM paměť . . . 14
2.4 Vstupně/Výstupní bloky . . . 15
2.5 Propojovací prvky . . . 15
2.6 Speciální funkční bloky . . . 15
2.7 Návrh programovatelných logických obvodů . . . 16
2.8 Jazyky HDL . . . 16
2.8.1 VHDL . . . 16
2.8.2 Verilog . . . 17
2.8.3 Ostatní jazyky . . . 17
2.9 Verifikace návrhu . . . 18

3 Rešerše na téma srovnání výrobců Altera a Xilinx **19**
3.1 Úvod . . . 19
3.2 Obecné srovnání . . . 19
3.3 Paměťové buňky . . . 20
3.4 Implementační nástroje . . . 22
3.5 Výkon procesorů . . . 22
3.6 Výhody Architektury Stratix III . . . 24

4 Rozbor možností logiky FPGA a periferních obvodů vývojového kitu Nios II pro použití v biomedicínské přístrojové technice **25**
4.1 Vývojový kit Nios II . . . 25
4.1.1 Základní deska kitu . . . 26
4.1.2 Multimediální deska kitu . . . 27
4.2 Využití v biomedicíně . . . 28
4.2.1 Monitorovací zařízení . . . 28
4.2.2 Implementace umělé neuronové sítě . . . 29

5 Seznámení se s vývojovým prostředím a zařízením Nios II **31**
5.1 Vývojové prostředí Quartus II 31
5.1.1 Qsys design suite 32
5.1.2 Pin planner 33
5.1.3 Build Tools For Eclipse 33
5.2 Projekt Hello from Nios II 33
5.2.1 Hardwarový návrh 34
5.2.2 Softwarový návrh 35

6 Návrh demonstrační úlohy s barevným maticovým LCD displejem **37**
6.1 Návrh zařízení 37
6.2 Rozbor úlohy 39
6.2.1 Hardwarová struktura 39
6.3 HW komponenty demo úlohy 40

7 Implementace demo úlohy do vývojového kitu Nios II **43**

8 Oživení demo úlohy **44**
8.1 Stav demo úlohy po spuštění 46
8.2 Analýza demo úlohy 47

9 Tvorba demonstrační úlohy **48**
9.1 Funkce využité v návrhu demonstrační úlohy 48
9.1.1 Inicializační funkce 49
9.1.2 Ovládací funkce 49
9.2 Sestrojení demonstrační úlohy 50
9.2.1 void EKG_init(int H, int V); 50
9.2.2 void Frame_init(void); 51

10 Zhodnocení dosažených výsledků **53**

11 Závěr **54**

12 Použitá literatura **55**

Seznam příloh na CD **58**

Seznam použitých zkratek

ALM	Adaptive Logic Model
BSP	Board Support Package
CD	Compact Disc
CE	Clock Enable
CPLD	Complex Programable Logic Device
EKG	Electrocardiography
EEG	Electroencephalography
FPGA	Field Programable Gate Array
FIFO	First In, First Out
HDL	Hardware Description Language
HSMC	High-speed Samtec Connector
IOB	Input/Output Block
LCD	Liquid Crystal Display
LUT	Look-Up Table
LB	Logical Block
NEEK	Nios II Embedded Evaluaiton Kit
PLL	Phase-Locked Loop
RAM	Random Acess Memory
SBT	Software Build Tool for Eclipse
TMSC	Taiwan Semiconductor Manufacturing Company
VHDL	Very High Speed Integrated Circuit Hardware Description Language

Seznam obrázků

1 Blokové schéma struktury obvodu FPGA 12
2 Obecný řez kombinace logických bloků 13
3 Vstupně/výstupní bloky FPGA 15
4 Příklad VHDL kódu 17
5 Příklad kódu ve verilogu 17
6 Simulace návrhu v simulátoru Model Sim 18
7 Základní architektura Xilinx Virtex-5 s přidruženou logikou 20
8 Základní architektura Altera Stratix III s přidruženou logikou 20
9 Porovnání implementačních nástrojů 22
10 Implementace pěti a tří vstupých funkcí 24
11 Vývojový kit NEEK, Cyclone III edition 25
12 Pohled na vývojový kit zespoda 26
13 Pohled na multimediální desku bez LCD 27
14 Architektura monitorovacího zařízení 29
15 Schéma modelu neuronu 30
16 Postup návrhu úlohy v jednotlivých částech vývojového prostředí 32
17 Součásti HW návrhu pro projekt Hello World 34
18 Návrh demonstrační úlohy 37
19 Blokové schéma LCD a jeho periferií 39
20 Hardwarové součásti úlohy 42
21 Nastavení kompilátorů 45
22 Vkládání zdrojových souborů 45
23 Srovnání skutečného výsledku s výsledkem na webových stránkách 46
24 Vývojový diagram demonstrační úlohy 48
25 Část funkce pro vykreslování demonstračního EKG signálu 50
26 Tělo funkce pro zobrazení rámečků a řetězce znaků 51
27 Grafický výstup demonstrační úlohy 52

Seznam tabulek

1 Srovnání platforem podle LUT a ALM 21
2 Srovnání výkonu podle rychlostního stupně 23

Seznam grafů

1 Testování využití LUT a ALM 21
2 Srovnání výkonu podle rychlostního stupně 23
3 Výkonová ztráta FPGA Virtex-5 24

1 Úvod

V dnešní době jde vývoj medicínské a biomedicínské techniky prudce kupředu, proto je nutné stále zdokonalovat řídící systémy, které zodpovídají za chod přístrojů či vyhodnocování výsledků. Díky vysokému výpočetnímu výkonu, možnosti rekonfigurace a cenové dostupnosti FPGA čipů se v tomto odvětví začínají používat čím dál víc častěji.

Na rozdíl od dříve používaných komplexně programovatelných logických obvodů či mikrokontrolérů, které zpracovávají instrukce sériově (postupně za sebou), hradlová pole jsou schopna pracovat paralelně (zpracovávat více procesů najednou), díky čemuž jsou FPGA čipy několikanásobně rychlejší. V některých případech mohou zařízení budit zdání, že zpracovávají instrukce paralelně, jedná se ale o multitasking, kdy procesor zpracovává jednotlivé procesy sériově za sebou vysokou rychlostí. Pro skutečné paralelní zpracování je nutné mít rozdělen procesor na více jader nebo využívat několik propojených zařízení.

Výstupem mnoha lékařských přístrojů pro vyšetřování jsou obrazová data, na jejichž základě lékař vyhodnocuje stav pacienta. Tato data jsou získávána v krátkých časových úsecích z několika úhlů a pro usnadnění práce lékařů, se tato data rekonstruují do 3D modelů, ve kterých je možno manipulovat s obrazem podle potřeby. Proces tvorby 3D zobrazení z jednotlivých řezů je velice náročný a zdlouhavý, proto se zde pro urychlení využívá schopnosti paralelního zpracování instrukcí.

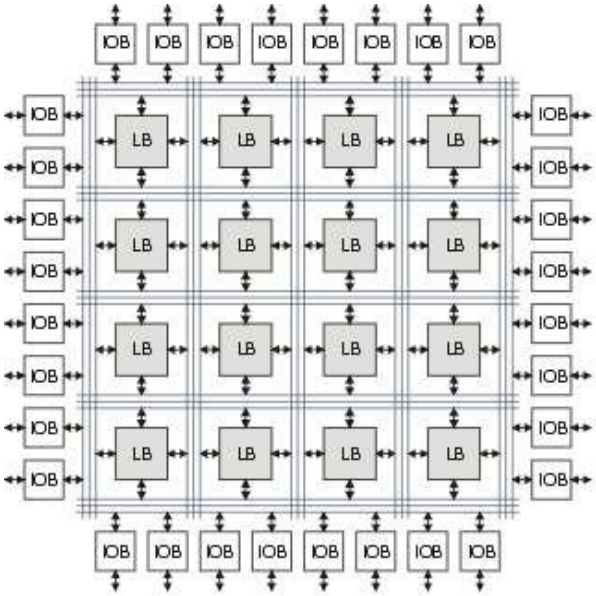
Na trhu s FPGA čipy si navzájem konkurují dva největší světoví prodejci hradlových polí, jedná se o výrobce Xilinx a Altera. Základní architektura čipů je u obou výrobců velice podobná, každý z nich se ale snaží o vytvoření zařízení, jehož výpočetní výkon bude co největší a využitelný na 100 % za každé situace.

2 Technika FPGA a návrh programovatelných logických obvodů

Současné obvody FPGA jsou složeny z několika milionů ekvivalentních hradel (AND, OR,...), které jsou systematicky uspořádány do logických buněk, jež zprostředkovávají jednoduché logické funkce. Tyto logické buňky je možné softwarově spojovat a vytvářet tak složitější a komplexnější obvod, pro jehož realizaci by bylo potřeba mnoho různých elektronických obvodů. Programovatelné hradlové pole se vyznačuje vysokou univerzálností, které je dosaženou velice obecnou strukturou obvodu, což vede k výraznému snížení potřeby integrovaných obvodů, jejichž funkce je předem dána výrobcem a nelze ji měnit.

2.1 Kombinační logika

Logické buňky jsou základními prvky FPGA, mají většinou 4 – 6 vstupů, z nichž jsou logické funkce vytvářeny pomocí struktury LUT (Look-Up Table). Tyto buňky se sdružují do logických bloků (LB – Logical Block). Používají se pro vytvoření funkcí a je možné je spojovat mezi sebou, díky čemuž můžeme vytvářet složitější funkce s více vstupy. Logické bloky obsahují krom logických buněk i další specializované prvky, sloužící pro jednodušší a efektivnější zapojení. Jedná se především o multiplexory, které jsou ve funkci spínače pro připojení logických buněk do dalších funkcí. Obvod také obsahuje vstupně / výstupní bloky (IOB – Input/Output Block), které ve spolupráci se zesilovači zajišťují komunikaci s okolím prostřednictvím logických napěťových úrovní. Většinou obsahují také klopné obvody, bloky pro správu hodinových signálů, bloky statických RAM pamětí a podobně.



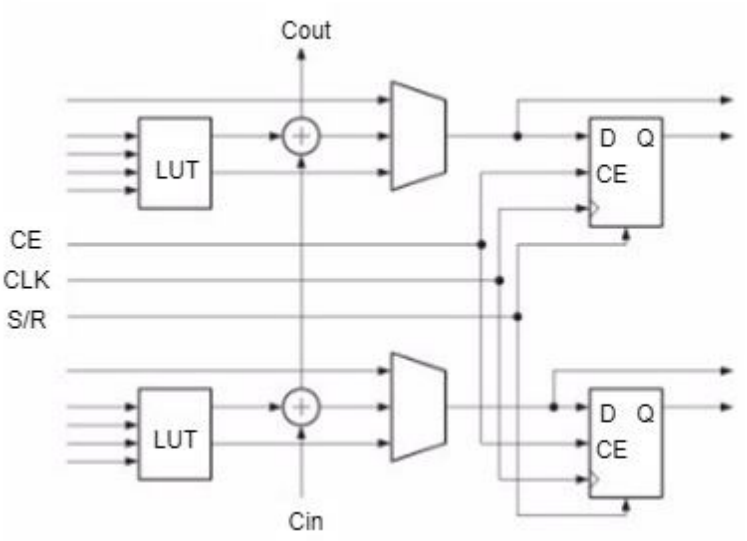
Obrázek 1: Blokové schéma struktury obvodu FPGA [2]

Na obrázku 1 můžeme vidět zjednodušenou blokovou strukturu obvodů programovatelných hradlových polí. Jednotlivé bloky je možné spojovat pomocí propojovací matice. Nežádoucím jevem při propojování logických bloků je vznikající zpoždění. Při propojování prostřednictvím

globální propojovací matice dochází ke zpoždění, které je v určitých případech možné eliminovat spojením některých signálů logických bloků se sousedním.

2.2 Logické bloky

Umožňují vytváření sekvenčních a logických funkcí s několika jednotkami popřípadě desítkami vstupů s několika výstupy.



Obrázek 2: Obecný řez kombinace logických bloků [4]

Konfigurovatelné logické bloky se většinou realizují kombinací dvou menších bloků, jejichž znázornění je viditelné na obrázku 2. V řezu se obvykle znázorňují tyto čtyři části:

- registry,
- multiplexory,
- programovatelné logické tabulky (LUT),
- řetězce rychlého šíření přenosu.

Registry zajišťují realizaci sekvenčních logických funkcí. Při konfiguraci řezů se společnými signály nastavují jednotlivé registry v rámci jednoho spojování bloku. Všechny používají stejný hodinový signál, ale díky signálu CE (Clock Enable) nemusí všechny registry tento signál používat.

Multiplexory realizují logické funkce s větším množstvím signálů, než dokáže samotná programovatelná logická tabulka řídit. Multiplexory nejsou nezbytnou součástí, ale zajišťují rychlejší přenos při spojování sousedních buněk.

Programovatelné logické tabulky (LUT – Look-Up Table) realizují libovolné kombinační logické funkce. V podstatě se jedná o paměti typu RAM, jejichž obsah je dán logickou hodnotou konfiguračních signálů. Z těchto tabulek můžeme také realizovat jiné funkční bloky například posuvné registry. Každá tabulka je schopna vytvořit 16 paměťových buněk.

Řetězec rychlého šíření přenosu se stará o vytváření aritmetických obvodů. Na rozdíl od běžných vstupů není připojen na pomalejší globální spojovací matici, ale signály CIN a COUT jsou připojeny přímo k sousednímu logickému bloku.

2.3 Realizace paměti RAM

V obvodech programovatelných hradlových polí nalezneme blokovou paměť, která má vyhrazené své místo a velikost výrobcem, ale můžeme si také vytvořit tzv. distribuovanou paměť pomocí řezů při kombinaci logických buněk.

Paměť RAM si můžeme představit jako několik klopných obvodů typu D, ke kterým se přistupuje prostřednictvím dekodéru, jehož výstupy jsou připojeny ke vstupům jednotlivých klopných obvodů. K zápisu dochází pomocí signálu clk. Čtení uložených dat je realizování pomocí multiplexoru, který na základě adres vybere požadovaný údaj.

2.3.1 Bloková RAM paměť

Vlastnosti pamětí závisí na výrobci a typu daného hradlového pole. Je vhodná pro realizaci složitějších paměťových podsystémů na hradlovém poli. Uspořádání bloků v podsystémech musí být stejné jako uspořádání bloků v paměti, což vysoce snižuje univerzálnost této paměti.

- **Jednoportová**

  Obsahuje jednu adresovou sběrnici a jeden vstup, který volí režimy mezi čtením a zápisem. Datová sběrnice může být obousměrná nebo oddělená (vstupní a výstupní datová sběrnice).

- **Částečně dvouportová**

  Jedná se o adresovou paměť, která je vybavena oddělenými datovými sběrnicemi pro čtení a zápis. Adresová sběrnice bývá taktéž oddělená.

- **Plně dvouportová**

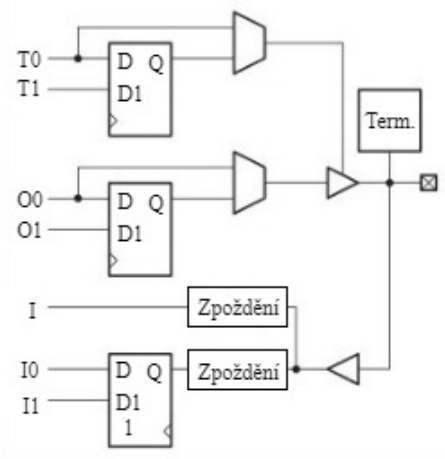
  Je ovládána pomocí dvou nezávislých sad řídících vývodů (datové, adresové a řídící), což umožňuje přístup dvěma různým obvodům ke stejné paměťové matici s různými požadavky. Pro zajištění správného výsledku je vhodné, aby se v jednu chvíli nepřistupovalo na jednu adresu s více různými požadavky.

2.3.2 Distribuovaná RAM paměť

Jedná se o paměti vytvořené z programovatelných logických tabulek zmíněných v 2.2, které pro svou funkci nepotřebují velké množství kombinační logiky jako například registry.

2.4 Vstupně/Výstupní bloky

Zajišťují připojení vnějšího prostředí k FPGA, zprostředkovává vstupní a výstupní signály.



Obrázek 3: Vstupně/výstupní bloky FPGA [4]

Jeho hlavním úkolem je přizpůsobení signálů různým napěťovým úrovním a obnovení časování signálu. V těchto blocích se obvykle vyskytuje několik oddělovacích zesilovačů a klopných obvodů.

2.5 Propojovací prvky

Vzhledem k tomu, že se od FPGA očekává vysoká flexibilita, je nutné mít dostatek dlouhých vodičů a spínačů pro propojení buněk. Bohužel každý spínač má parazitní jevy jako je kapacita nebo zpoždění, což ovlivňuje průchozí signál.

Propojovací vodiče jsou rozděleny podle jejich délky na vodiče jednoduché délky (propojují dva sousední bloky), vodiče dvojité délky (spojují bloky ob jeden) a dlouhé vodiče (mohou spojit celý řádek nebo sloupec matice logických bloků).

Programovatelné přepínače se používají pro připojení vstupů a výstupů nebo jako propojovací body v matici přepínačů, kterými se propojují jednotlivé segmenty propojovacích vodičů a vytvářejí tak sítě.

2.6 Speciální funkční bloky

FPGA čipy obsahují funkční bloky, které usnadňují jejich používání. Mezi tyto funkční bloky se řadí blokové paměti, aritmetické obvody, obvody pro správu a generování hodinových signálů atd.

Aritmetické obvody jsou většinou reprezentovány násobičkami popřípadě složitějšími bloky, umožňujícími provádění náročnějších operací v jednom hodinovém cyklu.

Blokové paměti mají obvykle kapacitu až několik desítek kilobytů a nastavitelnou šířku sběrnic. Obvykle se jedná o statické RAM paměti, ale mohou sloužit pro vytvoření jiných typů pamětí například FIFO.

Bloky pro správu a generování hodinových signálů nabízejí dělení hodinových signálů, jejich syntézu, generování fázově pozměněných hodinových signálů atd. Tyto bloky nelze jednoduše nahradit pomocí základní FPGA logiky, jsou připojeny ke speciálním bodům, které jsou přímo určeny pro šíření hodinových signálů. [1][2][3][4][7][8]

2.7 Návrh programovatelných logických obvodů

Pro přesný popis logických obvodů se už v historii používala schémata zapojení, která se s větší složitostí navrhovaných obvodů stávala nepřehlednýma. Proto bylo nutné najít jiný způsob, jak popsat logické obvody a jejich schémata. Výsledkem je textový popis velice podobný programovacím jazykům. Tento popis nese název HDL (Hardware Description Language).

Navzdory tomu, jak jsou si programovací jazyky a jazyky HDL podobné, není možné je zaměňovat. HDL jazyky definují strukturu logického obvodu. Zásadním rozdílem mezi programovacími jazyky a HDL jazyky je rozdíl v posloupnosti vykonávání instrukcí. Každá část popsaná v jazyce HDL je vykonávána pouze paralelně.

2.8 Jazyky HDL

V současné době se nejvíce používají jazyky VHDL a Verilog. Jazyky mají sice odlišnou syntaxi, ale oba slouží a poskytují popis logických obvodů. Jazyky je možno používat pro programování většiny hradlových polí, implementace struktury se pak v závislosti na výrobci liší kompilátorem.

2.8.1 VHDL

Jazyk vznikl z důvodu narůstající nereprodukovatelnosti elektronických systémů z důvodu vysoké složitosti dokumentace. Účelem vzniku byly široké popisovací schopnosti, byl kompatibilní s jakýmkoli simulátorem a nezávislý na vývoji technologií.

```
seg_on: process (clk,ted)
   begin
      if Rising_Edge (clk) then
         if flag = '1' then
            case move is
               when "00000000" => seg <= "11000000"; --0
               when "00000001" => seg <= "11111001"; --1
               when "00000010" => seg <= "10100100"; --2
               when "00000011" => seg <= "10110000"; --3
               when "00000100" => seg <= "10011001"; --4
               when "00000101" => seg <= "10010010"; --5
               when "00000110" => seg <= "10000010"; --6
               when "00000111" => seg <= "11111000"; --7
               when "00001000" => seg <= "10000000"; --8
               when others     => seg <= "10010000"; --9
            end case;
         end if;
      end if;
   end process seg_on;
```

Obrázek 4: Příklad VHDL kódu [4]

Díky standardizaci jazyka VHDL je také přenositelný a téměř nezávislý na cílové platformě, což umožňuje popis logického obvodu formou schématu, jehož výhodou je mnohem větší přehlednost, než při pohledu na kód. Návrhář může využívat systémovou knihovnu, která obsahuje několik stovek součástek rozdělených podle funkcí a tímto vytvořit hierarchickou strukturu obvodu.

2.8.2 Verilog

Tento standardizovaný jazyk pro popis číslicových obvodů vznikl spolu s číslicovým simulátorem Verilog-XL. Po několika letech byl jazyk spolu s autorskými právy přenesen pod neziskovou organizaci Open Verilog, kterou je nyní udržován.

```
module GCD_Calculator( clk, rst,start, P,Q,R,valid,State_Y);
 input clk, rst, start;
 input [7:0] P,Q;
 output [7:0]R;
 output valid;
 output [1:0] State_Y;

 wire valid,Ldp,Ldq,Ldr,Selp,Selq,Sela,Selb;
 wire eq, gth;

 //component instantiation Datapath
 Datapath DP (clk,rst,Ldp,Ldq,Ldr,Selp,Selq,Sela,Selb,P,Q,eq,gth,R);

 //componet instatntiation Control_Unit
Control_Unit CU( clk,rst,start, eq,gth, valid, Ldp,Ldq,Ldr,Selp,Selq,
 Sela,Selb,State_Y);

 endmodule
```

Obrázek 5: Příklad kódu ve verilogu [10]

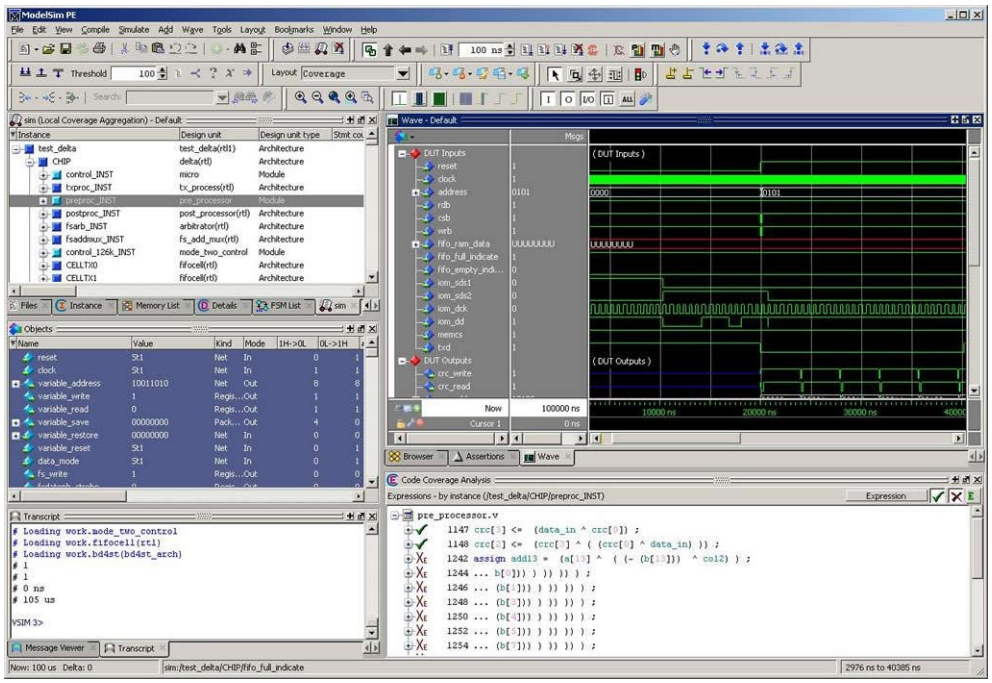
2.8.3 Ostatní jazyky

Kromě obvyklých HDL jazyků existují i jiné jazyky určené k popisu logických obvodů, mají ale vyšší úroveň abstrakce. Většinou se jedná o jazyky, které vznikly z jiných programovacích

jazyků, jedná se například o jazyk HLS od firmy Xilinx, který vychází z programovacího jazyka C.

2.9 Verifikace návrhu

Verifikace návrhu může probíhat za pomoci simulace nebo také diagnostikou. Obě tyto metody mohou být stejně časově náročné, jako samotná tvorba návrhu.



Obrázek 6: Simulace návrhu v simulátoru Model Sim [12]

Většina vývojových prostředí podporuje simulaci, která slouží k ověření správné funkce návrhu. Funkční simulace je rychlá a využívá ideálních vlastností použitých součástek. Tuto simulaci není vhodné používat ve složitějších návrzích, kde je možný výskyt různých logických hazardních stavů. V takovém případě se používá časová simulace, která již zahrnuje předpokládané parazitní jevy použitých součástek.

Pro diagnostiku reálného zapojení se používá logický analyzátor, jehož použití je bohužel omezeno pouze na signály, které jsou vyvedeny vně platformu FPGA. Programovatelná logika ale přináší možnost integrace diagnostických prostředků do navrhovaného systému. Takovýmto způsobem je možné využít zvolený datový kanál a získat informace o momentálním stavu uvnitř obvodu. [5][6][9]

3 Rešerše na téma srovnání výrobců Altera a Xilinx

3.1 Úvod

Práce se zabývá srovnáním FPGA od výrobců Altera a Xilinx. Cílem této rešerše je podat objektivní náhled na jednotlivé architektury a vytvořit soubor, v němž by potenciální uživatel mohl najít inspiraci pro volbu čipu pro realizace jeho projektu.

3.2 Obecné srovnání

Výrobci FPGA platforem Altera a Xilinx jsou jedni z nejznámějších a celou řadu let mezi nimi panuje velká rivalita.

Prodejní úspěšnost těchto platforem se příliš neliší. Xilinx ovládá přibližně 45% – 50% tržního limitu, zatímco Altera se svými 40 % - 45 % mírně pokulhává. Cenová dostupnost obou produktů je velice podobná, ovšem cena by neměla být základním parametrem, podle kterého se FPGA vybírá. Alteru je možné pořídit již od 251 Kč, zatímco poněkud dražší Xilinx je možno zakoupit od 303 Kč.

Před více než deseti lety používal Xilinx sadu nástrojů s názvem ISE, které se Altera dlouhou dobu nedokázala vyrovnat. Poté ovšem přišla se zbrusu novou sadou nástrojů Quartus, která skončila fiaskem, bohužel došlo i k odpuzení zákazníků. Altera si svou reputaci vylepšila vydáním vylepšené verze sady nástrojů Quartus II, která vedla společnost k velkému nástrojovému náskoku před Xilinxem. Quartus II je velice elegantní, intuitivní a připraven pro práci s inovovanými FPGA čipy.

Jako odpověď na Quartus II přišla firma Xilinix s vývojovými nástroji s názvem Vivado, jimiž se opět dostala do vedení. Tento balíček nástrojů je velice dobře koncipován, integrován a velice rychle se rozšiřuje. Jeho jedinou nevýhodou je raný věk a firma Xilinx musí ještě doladit některé vady.

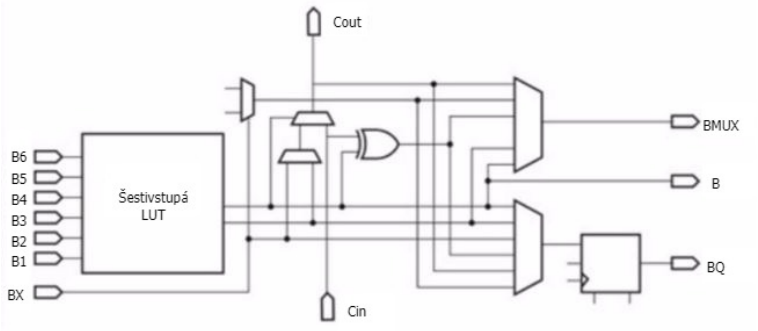
Společnost Altera je podle prodejních statistik považována za druhou, což budí dojem, že je jednodušší jí konkurovat, ale vzhledem k náročnosti výroby polovodičů není jednoduché konkurovat. Vývoj FPGA jde v obou společnostech kupředu a společnosti se navzájem předhánějí, díky čemuž může být chvíli konkurenceschopnější Altera a zanedlouho zase Xilinx.[13]

Návrháři pozorují rozdíly v navrhovacích nástrojích. Firma Xilinx v nových vydáních směřuje více k přidáváním různých tlačítek s funkcemi, zatímco Altera se snaží nabízet více informací pro optimalizaci výkonu návrhu. Architektura Altery je konstruována velice obecně a je jednoduše přizpůsobitelná jemným optimalizacím, na druhou stranu Xilinx se snaží být více technologicky orientovaný tím, že nabízí více čipů s vlastními obvody, které implementují speciální funkce. Společnost Altera podepsala smlouvu s TMSC (Taiwan Semiconductor Manufacturing

Company) o výrobě hradlových polí s hustotou integrace 14nm, čímž došlo ve firmě Intel k vytvoření čipu, který překonal čipy firmy Xilinx s hustotou integrace 16nm.[18][14]

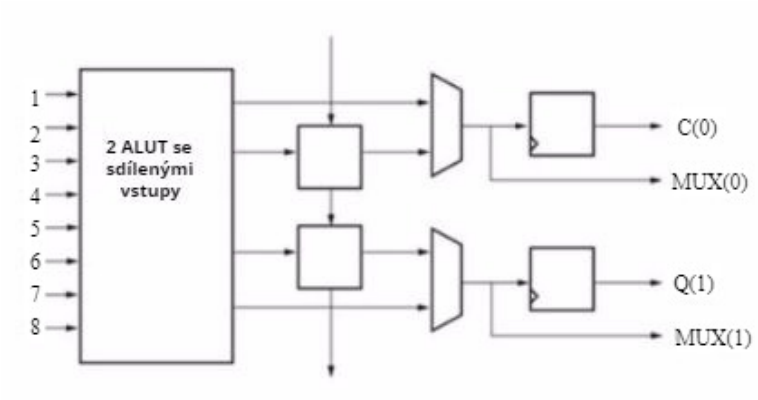
3.3 Paměťové buňky

Jak již bylo zmíněno v sekci 2.1, jedním ze základních stavebních prvků v architekturách FPGA jsou paměťové buňky LUT (Look Up Table), které se využívají pro realizaci logických funkcí. Konkrétní architektura se u každého z výrobců liší.



Obrázek 7: Základní architektura Xilinx Virtex-5 s přidruženou logikou [15]

Na obrázku 7 můžeme vidět architekturu Xilinx Virtex-5, která obsahuje LUT se šesti vstupy, které poskytují 64 bitový programovací prostor. Je schopen implementovat libovolnou funkci, popřípadě kombinaci jedné nebo dvou menších funkcí se šesti vstupy. V některých případech může být tato LUT použita jako 64 bitová RAM paměť.



Obrázek 8: Základní architektura Altera Stratix III s přidruženou logikou [15]

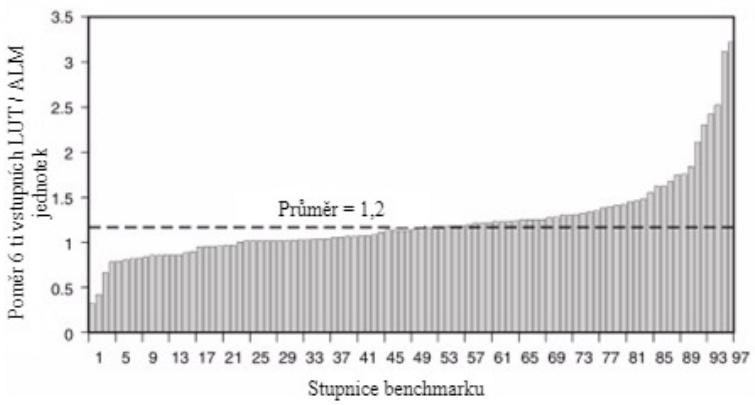
Na obrázku 8 můžeme vidět základní architekturu adaptivního logického modelu (ALM), který obsahuje dvě paměťové buňky poskytující 64 bitový programovací prostor s osmi sdílenými vstupy.

Adaptivní logický model poskytuje vysokou flexibilitu funkčnosti LUT se šesti vstupy a přidruženou logikou, nevýhodou je ale využívání větší křemíkové plochy. Srovnání ALM a LUT můžeme vidět v tabulce 1.

Tabulka 1: Srovnání platforem s ohledem na LUT a ALM [15]

Virtex-5	Šesti vstupé LUT	Stratix III	Počet ALM	Ekvivalentní počet LUT
XC5VLX20T	12480			
XC5VLX30 XC5VLX30T	19200	3SL50	19000	22800
XC5VLX50 XC5VLX50T	28800	3SL70	27000	32400
XC5VLX85 XC5VLX85T	51840	3SL110	42600	51120
XC5VLX110 XC5VLX110T	69120	3SL150	56800	68160

Tato analýza ukazuje, že zařízení Virtex-5 má o hodně více logické kapacity než zařízení Stratix III, což je jistě zapříčiněno taky tím, že ALM je oproti samotným LUT rozměrnější.



Graf 1: Testování využití LUT a ALM [15]

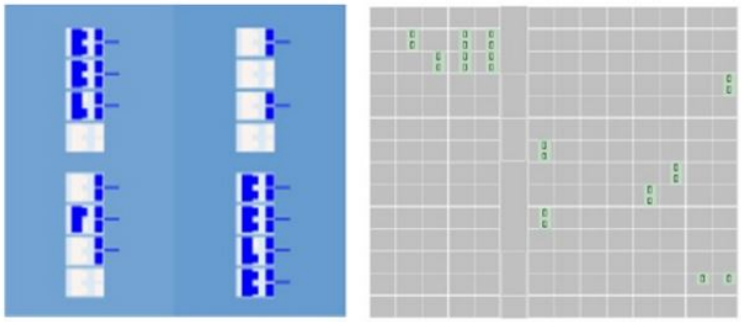
Na grafu 1 je osa x reprezentována průměrným počtem LUT se šesti vstupy, které jsou potřeba v ALM pro implementaci každého návrhu. Na ose y je stupnice seřazená od nejnižšího k nejvyššímu šesti vstupního poměru mezi LUT a ALM.

Výsledky jsou v rozsahu od 0,3 do 3,2 s průměrným výsledkem 1,2 LUT. Z testu je možné usoudit, že je potřeba 1,2 LUT pro realizace stejné logické funkce pomocí 1 ALM.

Další důležitou statistikou je využití LUT u Vritex-5 a ALM u Stratix III. Protože ALM obsahuje klopné obvody připojené na společné vstupy, je využita celá ALUT i pokud nejsou používány všechny její vstupy. Na rozdíl od toho vstupy u Virtex-5 FPGA zůstávají stále použitelné, protože jsou od sebe navzájem oddělené. [15]

3.4 Implementační nástroje

Každý z výrobců používá také své implementační nástroje, které nakládají s pamětí hradlových polí a umisťují data do volného paměťového prostoru, což má vliv na výkon FPGA.



Obrázek 9: Porovnání implementačních nástrojů [16]

Obrázky ukazují práci implementačních nástrojů jednotlivých platforem vlevo Stratix II, vpravo Virtex-5. Na platformě výrobce Xilinx vzal implementační nástroj data z prvních dvou klopných obvodů a umístil je na vstupu FPGA, ostatní data náhodně rozdělil po celém FPGA. Programovací nástroj firmy Altera rozdělil všechna data z klopných obvodů náhodně.

Programovací nástroj od společnosti Altera dává lepší umístění do pole klopných obvodů, protože jsou umístěny blízko u sebe, což ale nemusí být nutně spojeno s lepším výkonem. Časy šíření byly také proměnlivé a maximální frekvence pro správný provoz byla určena délkou nejdelšího signálu. Po ověření rychlosti příjmu dat bylo zjištěno, že s Virtex 4 můžeme dosáhnout rychlosti 30 MSPS, zatímco se Stratix II jen 11 MSPS.

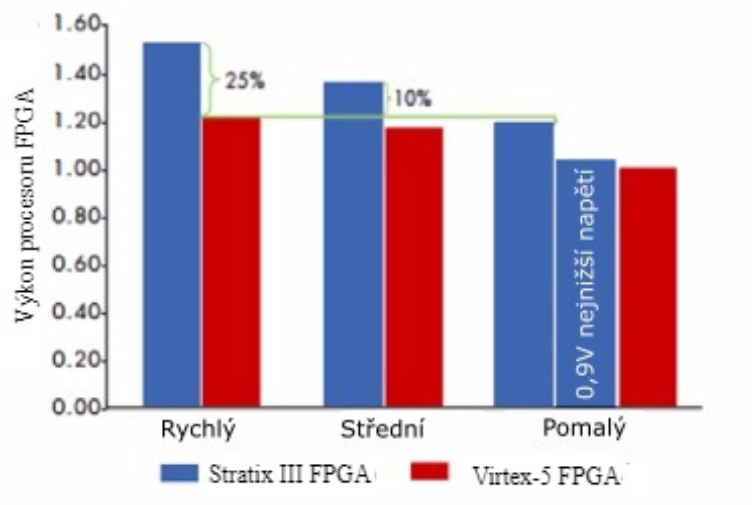
Pro zvýšení rychlosti komunikace byla použita funkce RLOC a editor přiřazení buněk klopných obvodů. Tímto je umožněno ruční umístění použitých mechanismů, což při správném nastavení může ovlivnit rychlost komunikace.

Virtex 4 po této změně dosáhl velikého zrychlení až na 90 MSPS, zatímco Stratix II dosáhl maximální rychlosti 15 MSPS. [16]

3.5 Výkon procesorů

Na základě používání nejnovějšího veřejně dostupného designového softwaru bylo při vysokém rychlostním stupni v průměru dosaženo o 25 % vyšší rychlosti než u konkurenčního 65nm zařízení (Xilinx Virtex-5) při stejném nastavení.

V důsledku větší výkonnosti se zařízení Stratix III při nízkém rychlostním stupni vyrovná konkurenčnímu zařízení Virtex-5 v rychlém rychlostním stupni.



Graf 2: Srovnání výkonu podle rychlostního stupně [17]

Tabulka 2: Srovnání výkonu podle rychlostního stupně [17]

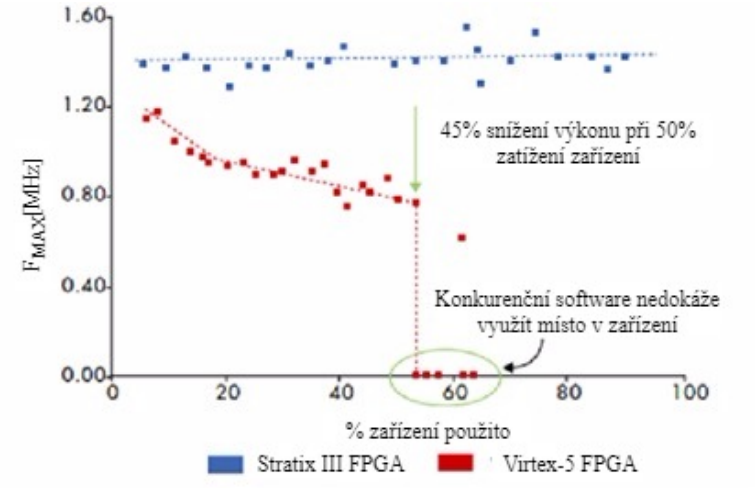
Rychlostní stupeň Stratix III	**Nejbližší rychlostní stupeň Virtex-5**	**Výhody ve výkonu Stratix III**
Rychlý	Není možné přiřadit	O 25 % rychlejší
Střední	Není možné přiřadit	O 10 % rychlejší
Pomalý	Rychlý	Shodný
0,9V nejnižší napětí	Pomalý	Shodný

Dále probíhalo testování maximální frekvence při postupném zatěžování FPGA. Výsledky tohoto testu ukazují:

- Software ISE Foundation není schopen přizpůsobit se verzi designu zaplňující více než 65 % zařízení,
- FPGA Virtex-5 prokázalo snížení výkonu o 45 % při plném zatížení tímto návrhem.

Pro dosažení co nejvyšší frekvence je nutné přizpůsobit vysokému výkonu nejen samotné jádro, ale také digitální bloky pro zpracování signálů a optimalizovat také okolní logiku, aby nedocházelo k omezování výkonu. Stratix III je schopen pracovat s rychlostí 600 MHz, zatímco Virtex-5 zvládá pouze 550 MHz.

FPGA Stratix III obsahuje paměť TriMatrix, která obsahuje bloky, jež jsou optimalizovány pro maximální možný výkon a umístěny do bloků, které je možné umisťovat kdekoliv v zařízení, což zajišťuje extrémní flexibilitu a podporu hodinového signálu nad 600 MHz.



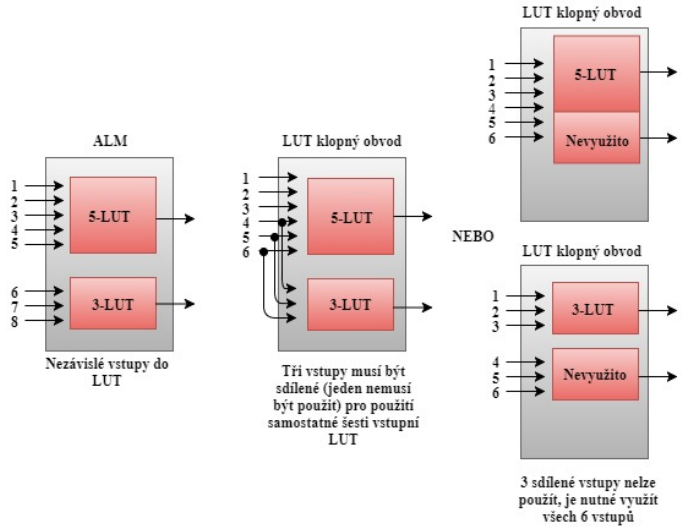
Graf 3: Výkonová ztráta FPGA Virtex-5 [17]

3.6 Výhody Architektury Stratix III

Klíčem k vysokému výkonu FPGA Stratix III je modul adaptivní logiky Adaptive Logic Modul (ALM), který je složen z kombinované logiky, dvou registrů a dvou přídavných prvků. Kombinovaná část má 8 vstupů a zahrnuje také vyhledávací tabulku Look-Up Table (LUT), kterou je možné rozdělit na dvě části.

Ve srovnání s FPGA Virtex-5 dosahuje Stratix III mnohem vyšší flexibility, protože Virtex-5 je omezen LUT se šesti vstupy, což sice stačí pro provádění dvou menších logických funkcí, ale v důsledku omezeného počtu vstupů bude ALM většinou používán jako šesti vstupní.

Kombinace osmi vstupů může efektivně implementovat dvě nezávislé funkce, které mohou vyžadovat klopné obvody, aniž by bylo nutné používat sdílené vstupy nebo zdroje. [17]



Obrázek 10: Implementace pěti a tří vstupých funkcí [17]

4 Rozbor možností logiky FPGA a periferních obvodů vývojového kitu Nios II pro použití v biomedicínské přístrojové technice

Řídící FPGA čip vývojového kitu nabízí 24 624 logických buněk, pomocí kterých je možné zpracovávat logické funkce a dále disponuje 216 fyzickými piny, které je možné ovládat. Tato čísla svědčí o široké využitelnosti tohoto vývojového kitu.

4.1 Vývojový kit Nios II

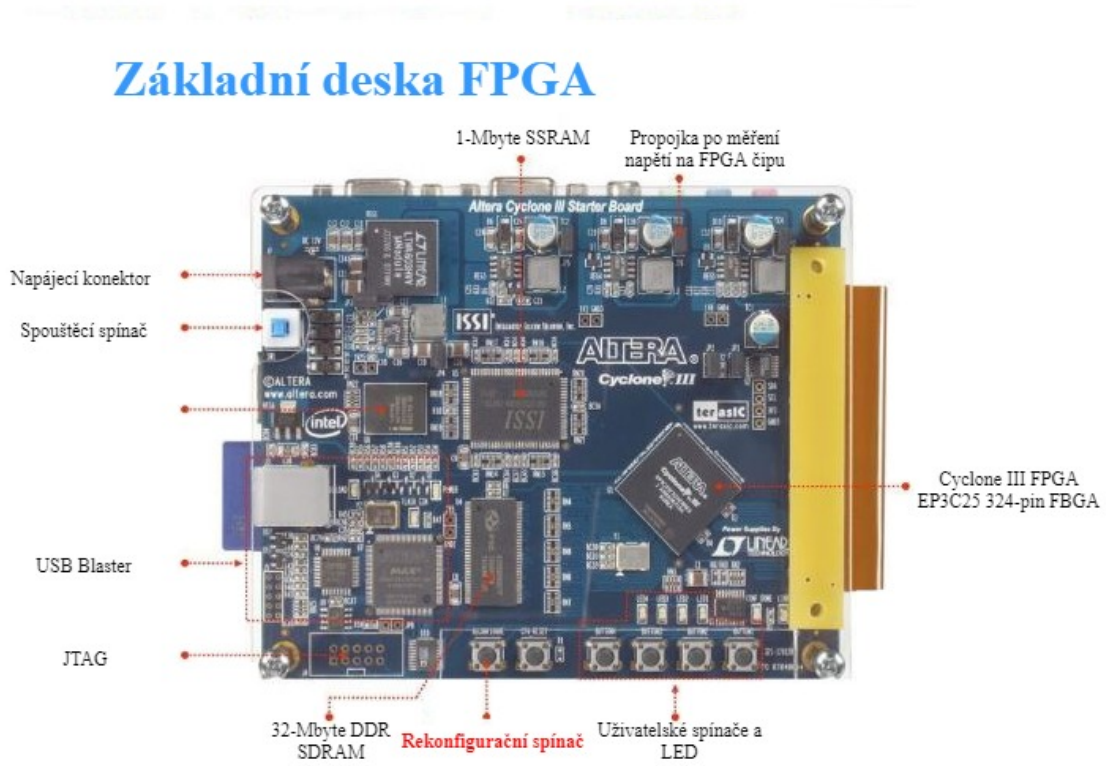
Tento vývojový kit obsahuje nízkonapěťový FPGA čip, kterým je za pomoci ostatních periferních obvodů možné nahradit až desítky procesorových systémů. Model NEEK, Cyclone III edition nabízí díky velkému počtu periferií rozsáhlé možnosti využití.

Obrázek 11: Vývojový kit NEEK, Cyclone III edition [20]

Z obrázku 11 je patrné, že kit je složen ze dvou různých desek, které jsou mezi sebou propojeny prostřednictvím HSMC konektoru.

4.1.1 Základní deska kitu

Nejdůležitějšími částmi základní desky jsou FPGA čip, paměti a konektor zajišťující propojení této desky s deskou multimediální.

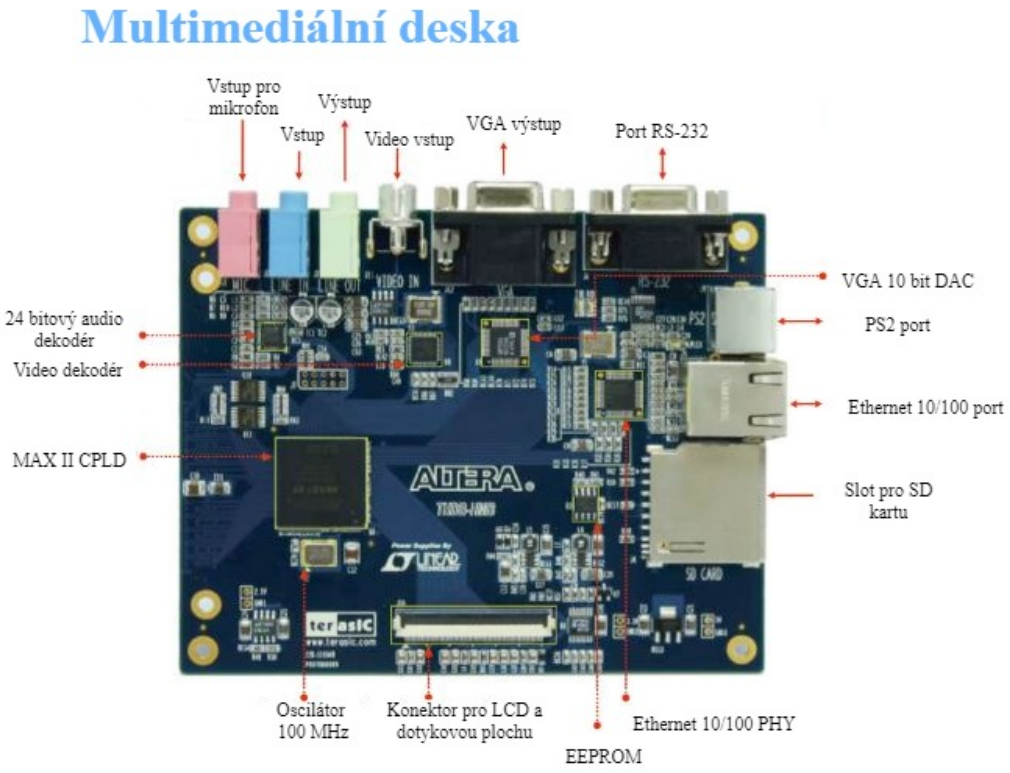


Obrázek 12: Pohled na vývojový kit zespoda [21]

Při pohledu na obrázek 12 můžeme vidět, že tato deska obsahuje výhradně obvody pro programování čipu, ovládání paměti, napájení a komunikaci s multimediální deskou. Uživatel zde může využít programovatelných spínačů a LED, se kterými je možné si procvičit realizaci úlohy v tomto zařízení.

4.1.2 Multimediální deska kitu

Tato deska obsahuje celou řadu periferií, které nabízí širokou škálu možností využití.



Obrázek 13: Pohled na multimediální desku bez LCD [21]

Zde již není umístěn žádný FPGA čip, ale je zde možné nalézt CPLD MAX II, které zpracovává signály pro LCD a upravuje napěťovou logiku ze 2,5V (napěťová úroveň na základní desce) na 3,3V (napětí potřebné pro správnou funkci periferních obvodů na multimediální desce).

Díky audio dekodéru je možné zpracovávat audio signál přijatý jedním ze dvou audio vstupů, z nichž jeden je určen pro připojení mikrofonu. Zpracovaný signál může být odesílán do periferních zařízení za použití výstupního konektoru.

Na desce může uživatel najít také konektor pro vstup video signálu a spolu s digitálně analogovým převodníkem také VGA výstup, pomocí kterého je možné zobrazovat data na počítačovém monitoru.

Další periferií je konektor RS-232, nabízející odesílání dat jinému zařízení pomocí sériové komunikace. Na desce je také obsažen port pro připojení ethernet kabelu, pomocí kterého je rovněž možné odesílat data.

Připojení počítačové klávesnice či myši je možné díky portu PS2.

Nedílnou součástí je také slot pro SD kartu, na kterou je možné ukládat projekty a spouštět je bez připojení počítače.

4.2 Využití v biomedicíně

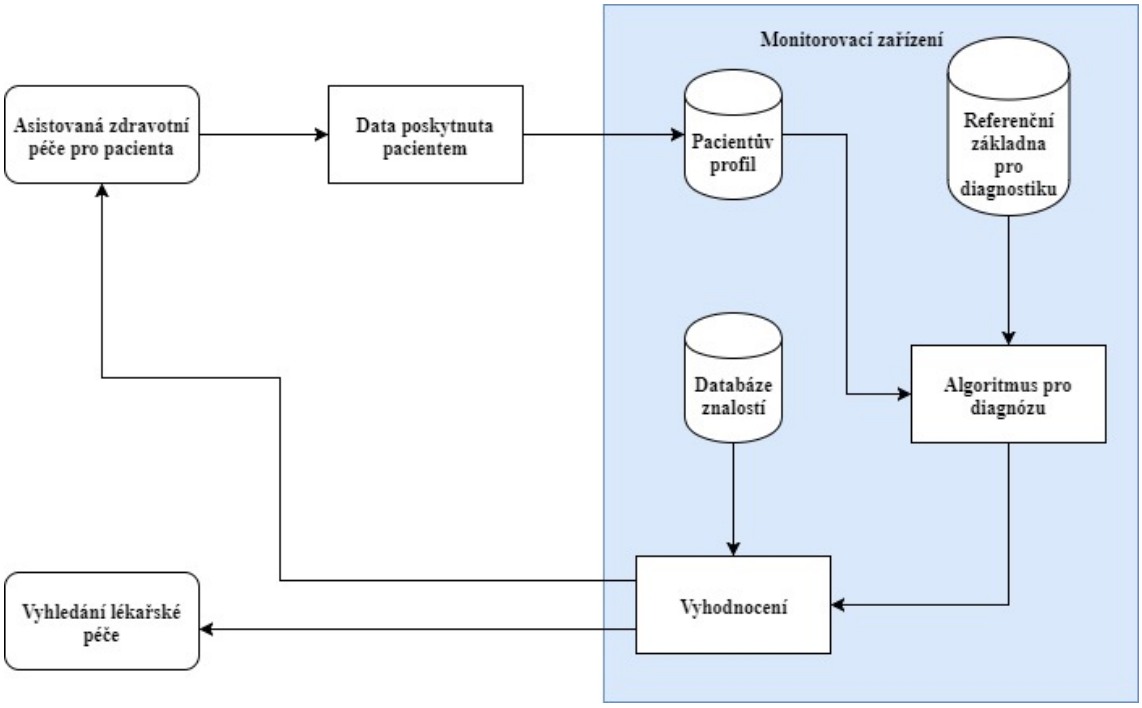
Biomedicínský obor se zabývá především využitím přírodovědeckých principů v medicíně, ale také lékařskou technikou a její inovací. Proto se někteří programátoři hradlových polí soustředí na spojení medicíny s vysoce výkonými programovatelnými čipy FPGA.

4.2.1 Monitorovací zařízení

Klinické rozhodování je komplikovaný proces, vycházející z lékařských znalostí, které pocházejících z knih, literatury a různých klinických či diagnostických testů. V některých zemích žije ve venkovských oblastech obrovský počet lidí, kterým není možné zajistit potřebnou lékařskou péči z důvodu nerovnováhy poměrů pacient/lékař. Například v Indických venkovských oblastech připadá na jednoho lékaře 10 000 pacientů, proto není možné zajistit všem obyvatelům potřebnou zdravotní péči a předcházet tak komplikacím.

S rostoucím vědomím o nutné zdravotní péči a zvyšujících se nákladech je v tomhle ohledu zvyšován nárok na lékařskou techniku. Požadavkem je levná, ale za to spolehlivá technika, se kterou budou moci lékaři manipulovat podle potřeby. Hradlová pole disponují velkým výpočetním výkonem, malými rozměry a, v poměru cena:výkon, levnou pořizovací cenou, proto se v tomto oboru vyskytují stále častěji.

FPGA čipy je možné využívat například pro výrobu zařízení, která budou monitorovat základní životní funkce pacientů, ukládat zjištěné hodnoty a kontrolovat, zda jsou v normě nebo ne. Zařízení uvědomí pacienta v případě zhoršení zdravotního stavu a na tomto základě by pacient měl neprodleně vyhledat lékařskou pomoc. Lékař bude schopen diagnostikovat nemoc po stažení dat z monitorujícího zařízení a tím je usnadněn proces dalších vyšetření či pozorování pacienta.



Obrázek 14: Architektura monitorovacího zařízení [22]

- Zařízení zachycuje údaje měřením zdravotních parametrů pacienta,
- lékař interaguje s monitorovacím zařízením a potvrzuje nebo odmítá diagnózu inteligentním nástrojem,
- monitorovací zařízení provádí diagnostiku v pravidelných časech a předpovídá budoucí stavy za pomocí fuzzy logiky.

Na základě naměřených hodnot může zařízení poskytnout včasný signál o zhoršení zdravotního stavu pacienta. Údaje z pacientova profilu jsou po porovnání s referenčními hodnotami podrobeny diagnostickému testování a porovnáním s databází. Inteligentní systém poskytuje indikaci možného dalšího fyziologického stavu pacienta. Diagnostikovat poruchu z jednoho datového toku by bylo velice obtížné a rozhodnutí by nemuselo být vždy pravdivé, proto se zde zavádí fuzzy logika, díky které je možné posuzovat, jak velká je pravděpodobnost, že k nějaké z poruch došlo. [22]

4.2.2 Implementace umělé neuronové sítě

„Umělá neuronová síť (UNS) si bere vzor v biologické neuronové síti, kterou má každý živý organismus. Při návrhu UNS se vychází z neurofyziologických poznatků, jako je struktura, uspořádání, stavební elementy nebo funkce. UNS modelují chování a činnost biologické neuronové sítě.“ [23]

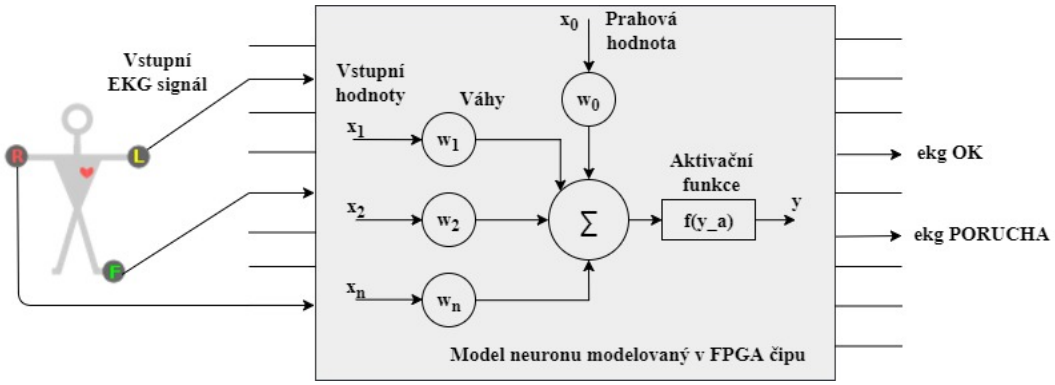
Takovouto neuronovou síť je možné využít například pro řízení strojů, detekci a vyhodnocení vzniklé poruchy či vyhodnocování EEG a EKG signálů. UNS je možné využít v případech, kdy

se nalezení algoritmu sledovaného procesu zdá být kvůli své složitosti téměř nemožné.

Největší výhodou těchto sítí je jejich schopnost učit se z připravených příkladů, díky čemuž jsou následně schopny různě reagovat na změny. Velikost odchylky o kterou se mohou vstupní data měnit je závislá na velikosti UNS, délce učení a počtu trénovacích množin.

Vzhledem k vysoké náročnosti je výhodné modelovat tyto sítě na FPGA čipech, protože je možné využít paralelního zpracovávání instrukcí, které může při správném návrhu zajistit vysokou efektivitu využití výpočetního výkonu.

Základním stavebním blokem těchto sítí jsou neurony, které jsou následně spojovány do sítí tvořených několika vrstvami. Neexistuje žádný přesný návod, ze kterého by bylo jednoznačné, jak tyto sítě správně tvořit, ale je možné inspirovat se různými doporučeními.



Obrázek 15: Schéma modelu neuronu [23]

Na obrázku 15 je možné vidět měření tří svodového EKG vstupujícího do FPGA čipu, ve kterém je namodelovaná neuronová síť (pro přehlednost obrázku je zobrazen model pouze jednoho neuronu), jejíž funkcí je vyhodnocovat vstupní signál.

Vstupem na každém takovém modelu je vektor hodnot X, ve kterém jsou promítnuty vstupní hodnoty neuronové sítě nebo výstupy předchozích neuronů v síti. Každému vstupu je také přiřazena jeho odpovídající váha symbolizující důležitost vstupu při následné sumarizaci. Ve chvíli, kdy součet všech vstupů převýší uživatelem nastavenou prahovou jednotku neuronu, dochází k aktivaci výstupu.

Pro využití v biomedicíně, lze tyto sítě naučit například vhodný průběh EKG či EEG signálu a porovnávat signál se signálem pacienta. Systém detekuje rozdíly na jejichž základě diagnostikuje poruchu.[23]

5 Seznámení se s vývojovým prostředím a zařízením Nios II

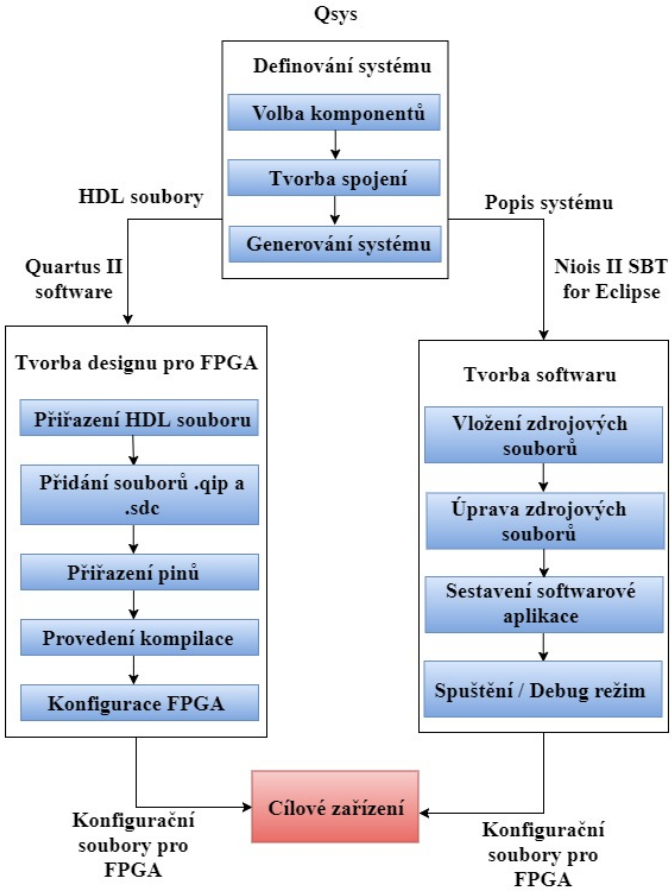
Před samotnou tvorbou demonstrační úlohy je potřeba naučit se pracovat s vývojovým kitem a vyzkoušet si realizaci jednoduché úlohy.

5.1 Vývojové prostředí Quartus II

Software Altera Quartus II je velice komplexní vývojové prostředí, pomocí kterého se provádí návrh logických zařízení firmy Altera.Tento software umožňuje provést časovou analýzu návrhů, simulovat reakci návrhu na různé podněty a nakonfigurovat cílové zařízení prostřednictvím programátoru. Quartus podporuje implementaci v popisových jazycích jak VHDL, tak i Verilog.

Software může být licencován jako web edition (nezpoplatněná verze) nebo subscription edition (zpoplatněná verze). Pro realizaci tohoto projektu je dostačující verze web edition, protože poskytuje veškeré potřebné funkce pro realizaci úlohy. Software s licencí web edition na rozdíl od subscription edition nepodporuje všechny FPGA čipy a není schopen zajistit implementaci do multiprocesorů. Má také omezenou funkčnost simulátoru.

Quartus II a většina jeho součástí se používá pro návrh v jazyce VHDL nebo Verilog, existuje ale také neopomenutelná součást s názvem Software Build Tools for Eclipse, která se využívá pro tvorbu kódu v jazyce C/C++, sloužící k ovládání navrženého hardwaru.



Obrázek 16: Postup návrhu úlohy v jednotlivých částech vývojového prostředí [24]

Návrh každé úlohy začíná v podprogramu Qsys, ve kterém se vytváří HW struktura spojením jednotlivých vložených, či uživatelem vytvořených komponent. Dalším krokem je generování systému a přesun do vývojového prostředí Quartus II, ve kterém se do programu přidávají další soubory a provádí přiřazení vývodů k reálným pinům FPGA. Následně prochází návrh kompilací a nahrává se do samotného zařízení. Poté se přechází do podprogramu Nios II SBT for Eclipse, kde se vytvářejí zdrojové soubory a ladí se navržený program. [24]

5.1.1 Qsys design suite

Jedná se o součást vývojového prostředí Quartus II a slouží k automatizaci při připojování komponent mezi sebou a sběrnicí, což vede k vytvoření kompletního modulu, který funguje na zvoleném typu FPGA. Qsys obsahuje knihovnu s předem připravenými komponentami, které je možné vkládat do návrhu a následně je konfigurovat. Uživatel si také může navrhnout svou vlastní komponentu.

Propojení mezi jednotlivými komponentami je zajištěno pomocí sběrnice Avalon, jejíž parametry jsou automaticky zpracovávány při generování navrženého systému.

Toto je jediné rozhraní, které se používá pro tvorbu modulu, obsahujícího jednotlivé kompo-

nenty (které mají často mnoho nastavitelných parametrů) a pro určení topologie sběrnice.

Výstupem tohoto podprogramu je soubor s příponou .sys obsahující „virtuální“ systém, který se poté přes programátor nahrává do zařízení.

5.1.2 Pin planner

Pin planner je další neodmyslitelnou součástí prostředí Quartus II, do které se programátor přesouvá po vytvoření návrhu hardwaru, aby přidělil jednotlivým komponentám vhodné piny z FPGA.

Jednotlivé piny zvoleného FPGA jsou graficky zobrazeny v okně. Pod nimi jsou vypsány piny, které je potřeba přiřadit. V případě potřeby je možné také zobrazit schéma zapojení každého z vývodů nebo modifikovat jejich parametry.

5.1.3 Build Tools For Eclipse

Další část programovacího prostředí, která se využívá k vytváření zdrojových kódů v jazycích C/C++. Jedná se o poslední nezbytné programovací prostředí v návrhu veškerých úloh. Výstupem jsou dva soubory. Soubor s příponou .bsp, vzniklý v závislosti na navrženém systému v Qsys a soubor .elf, který se generuje po kontrole kódu a následně se nahrává do zařízení.

Projekty je možné spouštět v normálním režimu, kdy se program nahraje do FPGA a bez dalšího přerušování se spustí. Další možností je debug režim, ve kterém je možné program krokovat pomocí známých funkcí step into nebo step over.

V nastavení těchto projektů je možné vkládat soubory (např. fotky) jako přílohy projektů, které budou následně zpracovávány v FPGA.

5.2 Projekt Hello from Nios II

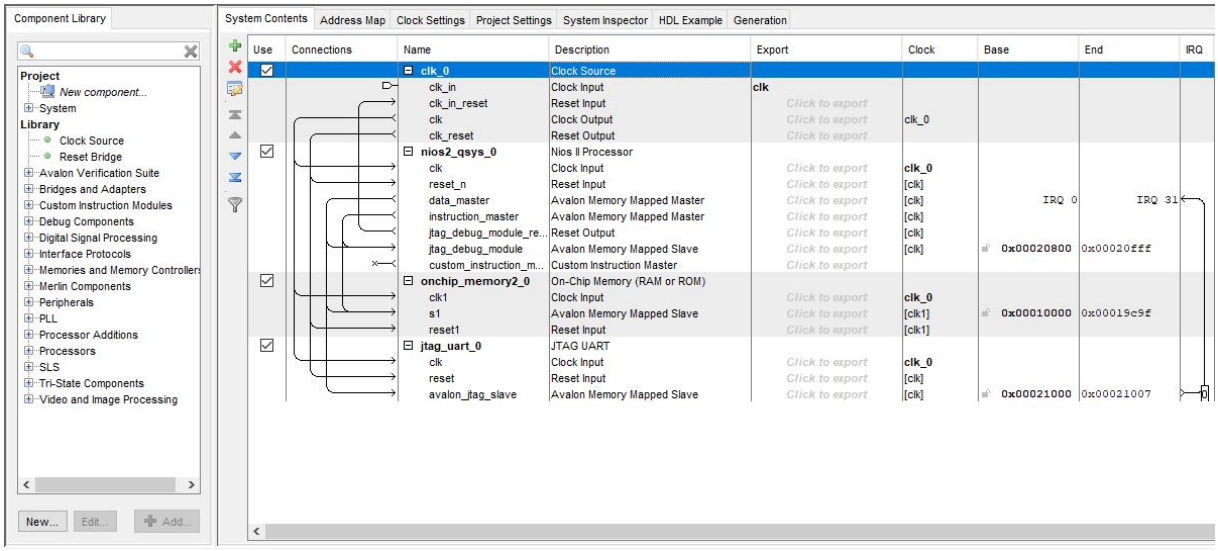
Tvorba tohoto projektu patří k jednomu z prvních kroků začínajících návrhářů. Uživatel je v průběhu návrhu seznámen se základními požadavky systému pro vytvoření jednoduché úlohy, jejíž závěrem je výpis řetězce „Hello from Nios II!“ do konzole.

Prvotním krokem při tvorbě všech návrhů je tvorba projektu ve vývojovém prostředí Quartus II. Zde je nutné zadat název projektu, místo kde bude uložen a zvolit typ používaného FPGA čipu.

V dalším kroku se pomocí záložky „Tools“ přechází do podprogramu Qsys, ve kterém se vytváří HW struktura návrhu. Výstupem tohoto podprogramu je mimo jiné také soubor s příponou .sopcinfo, který je dále využíván pro tvorbu dalších projektů v prostředí Eclipse.

5.2.1 Hardwarový návrh

Po vstupu do podprogramu Qsys ve vývojovém prostředí Quartus II se uživatel dostává na základní obrazovku. Na levé straně se nachází knihovna s jednotlivými komponentami, které se využívají při realizaci návrhu.



Obrázek 17: Součásti HW návrhu pro projekt Hello World

Na obrázku 17 jsou viditelné komponenty, které je nutno přidat do návrhu a propojit podle potřeby pro správnou funkci navrženého hardwaru. V návrhu jsou uvedeny základní adresy, ke kterým se přistupuje. Tyto adresy, spolu s resetovací globální sítí a prioritou přerušení, byly automaticky přiřazeny použitím funkcí „**Assign Base Adresses**“, „**Assign Interrupt Numbers**“ a „**Create Global Reset Network**“.

Posledním krokem je přepnutí se do záložky „**Generate**“, ve které dochází k vytvoření souboru .qsys.

Po návratu do prostředí Quartus II se v levé části okna nachází okno s názvem „Project Navigator“, kde je nutné přepnout záložku na „Files“ a vložit sem vytvořený soubor s příponou .qip.

Protože je v HW návrhu použit signál clk, je nutné přiřadit k němu správný pin čipu FPGA, na kterém je generován hodinový signál. V tomto případě se jedná o pin V9 a je možné jej přiřadit opět v záložce „Tools“ a podprogramu „**Pin planner**“.

Poté je nutné pro kontrolu provést kompilaci navrženého systému.

V záložce „Tools“ vývojového prostředí Quartus II se nachází podprogram „Programmer“, zde je nutné zvolit připojené zařízení a vytvořený konfigurační soubor .qsys.

Při připojování čipu FPGA k hostitelskému počítači často dochází k problému s ovladači USB blasteru. Ovladač je nainstalován spolu s vývojovým prostředím Quartus II a v tomto adresáři, konkrétně ve složce quartus a následně drivers, je možné najít potřebné ovladače, jejichž adresář je nutné manuálně zadat k zařízení USB blaster. Toho je možné docílit ve správci zařízení hostitelského počítače. Po připojení FPGA čipu, uživatel vidí v horní části správce zařízení připojený USB blaster, jehož ovladač je nutné manuálně aktualizovat tak, že vybereme možnost instalace ovladače z adresáře v zařízení a nastavíme dříve zmíněný adresář obsahující ovladač.

Pokud je na hostitelském počítači operační systém novější než Windows 7, je pravděpodobné, že se uživatel bude potýkat s problémy týkajících se dříve zmíněného ovladače, protože Quartus II verze 11.0 není zcela kompatibilní s tímto OS. Po každé systémové aktualizaci je zapotřebí znovu nastavit cestu k adresáři ovladače. Protože se jedná o starší verzi vývojového prostředí, ovladače neobsahují digitální podpis, který je při instalaci ovladačů vyžadován a každý ovladač je před instalací kontrolován systémem. Tuto kontrolu lze ale vypnout mnoha způsoby. Pro jednorázové vypnutí této kontroly, stačí stisknout klávesu shift a v nabídce start kliknout na restartovat. Systém se spustí s rozšířenými možnostmi spouštění, kde uživatel volí položku „Odstranit potíže“, následně „Nastavení spouštění“ a poté je uživateli nabídnut seznam možných jednorázových nastavení pro spuštění OS. Zde je možné zvolit zakázání vynucení podpisu ovladače pomocí klávesy F7. Poté se počítač spustí a je možné instalovat ovladače.

Po nahrání konfiguračního souboru do FPGA , pokračuje návrh v softwarové části za použití programu SBT. Pokud uživatel používá novější OS než Windows 7, je nutné nastavit kompatibilitu spouštění, tím je možné předejít dalším komplikacím. Kompatibilitu může uživatel nastavit ve vlastnostech programu, pravým kliknutím na ikonu SBT a vybráním políčka „Vlastnosti“ se otevře okno, obsahující záložku „Kompatibilita“, zde je nutné zaškrtnout políčko „Tento program spustit v režimu kompatibility pro:“ a následně zvolit Windows 7 popřípadě Vista service pack 2.

Při spouštění tohoto prostředí je nutné zvolit úložiště, ve kterém je uložený projekt, aby bylo možné vytvořit další součásti návrhu.

5.2.2 Softwarový návrh

Po otevření dříve zmíněného vývojového prostředí je nutné vytvořit 2 projekty:

- **BSP projekt**
  Při zakládání tohoto projektu je potřeba zvolit „**Nios II Board Support Package**“, vytvořit název projektu (pro přehlednost se v tomto projektu přidává přípona _bsp), vybrat soubor s příponou .sopcinfo z adresáře s projektem, ostatní možnosti zůstávají nezměněny,

  Zde jsou obsaženy základní ovladače pro FPGA, které uživatel neprogramuje, ale jsou

vytvořeny autonomně v závislosti na využitých periferiích.

- **APP projekt**
  Tentokrát se při zakládání volí „**Nios II Application from BSP Template**“, definuje se opět název (s příponou _app) a zadává se cesta k dříve použitému .sopcinfo souboru. Jako vzor pro tento projekt stačí ponechat výchozí zvolený Hello World, který zajistí vytvoření souboru .c se základním příkazem pro výpis řetězce znaků do konzole.

Nyní je možné otevřít soubor HelloWorld.c, ve kterém je viditelný příkaz „**printf(”Hello from Nios II!”);**“. Tento příkaz vypíše do konzole programu Eclipse řetězec uvedený v uvozovkách pomocí příkazu printf.

Pro implementaci kompletního systému do FPGA čipu je nutné vytvořit BSP soubor, čehož je možné dosáhnout při kliknutí na projekt s příponou _bsp pravým tlačítkem myši, poté se přechází na záložku „Nios II“ a zde se volí „**Generate BSP**“.

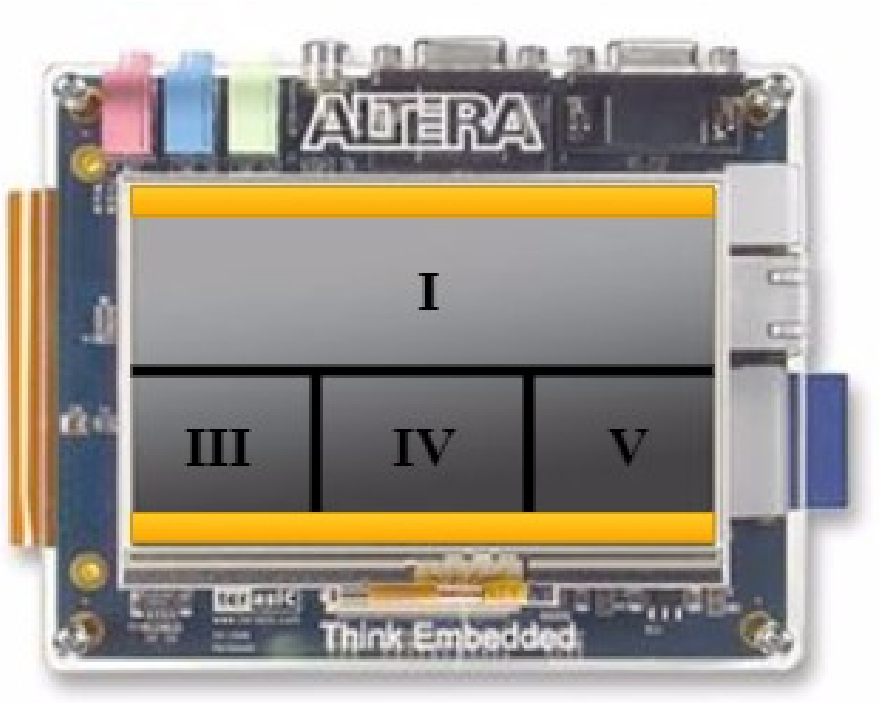
Dalším a posledním krokem je nahrání vytvořené aplikace do FPGA, což je možné provést kliknutím na projekt s příponou _app pravým tlačítkem myši, následným přechodem na záložku „Run As“ a volbou „**Nios II Hardware**“.

6 Návrh demonstrační úlohy s barevným maticovým LCD displejem

Po seznámení se s vývojovým prostředím a oživení kitu je možné přejít k návrhu samotné demonstrační úlohy.

6.1 Návrh zařízení

Tato úloha bude simulovat možné rozložení LCD pro plnění funkce přenosného monitoru životních funkcí. Zobrazená čísla a signály budou ideální a pouze demonstrační. Návrh tohoto monitoru životních funkcí bude realizován na barevném maticovém LCD, s rozdělením dle následujícího obrázku:



Obrázek 18: Návrh demonstrační úlohy[21]

Obrázek 18 jasně definuje rozdělení displeje za pomocí římských číslic. LCD má rozlišení 480 x 800 pixelů, proto se umístění jednotlivých bloků bude odvíjet právě od toho rozlišení. Souřadnice jsou přesně definovány pro osu x a y.

Jednotlivé části displeje budou zpracovány následujícím způsobem:

- Žlutý rámeček nahoře bude vyhrazen pro typ závěrečné práce s rokem realizace práce, studijním oborem a osobním číslem studenta. Tyto údaje budou vypsány v rámečku o rozměrech 50 x 800 pixelů se zeleným pozadím.
- Prostor označen číslem I je vyhrazen pro vykreslení dvou cyklů ideálního EKG signálu červenou barvou na černém pozadí. Pro zobrazení tohoto signálu je vyhrazena část displeje o rozměrech 230 x 800 pixelů.

- Čísly II, III a IV jsou označena místa, ve kterých bude zobrazen prostor pro zobrazení hodnot dechové frekvence, krevní saturace kyslíkem a krevního tlaku. Jednotlivé parametry budou zobrazeny červenou barvou ve stejně velkých tmavě modrých rámečcích vedle sebe o rozměrech 149 x 266 pixelů. Tyto rozměry jsou definovány tak, aby zůstaly volné 2 pixely na šířce displeje, které budou sloužit pro oddělení jednotlivých bloků od sebe.
- Žlutý rámeček ve spodní části LCD s rozměry 50 x 800 pixelů je vymezen pro výpis názvu vývojového kitu, na kterém je tato demonstrační úloha realizována.

Pro zobrazení dat v textovém formátu bude využito tučného písma fontu Tahoma v černé barvě.

Tato demonstrační úloha bude založena na demo úloze „Video and Image Processing Suite Demo on the NEEK“, kterou je nutno rozebrat a upravit podle dříve zmíněného zadání.

Demonstrační úloha bude zpracovávána ve vývojovém prostředí Quartus II verze 11.0 firmy Altera. Vývoj HW návrhu bude probíhat v programu Qsys, ve kterém bude vytvořen soubor obsahující všechny komponenty potřebné pro oživení LCD a komunikaci s digitizérem.

V rámci vývojového prostředí Quartus II budou k vytvořenému modelu, obsahujícím veškeré komponenty, přiřazeny vhodné piny FPGA za použití nástroje „Pin Planner“.

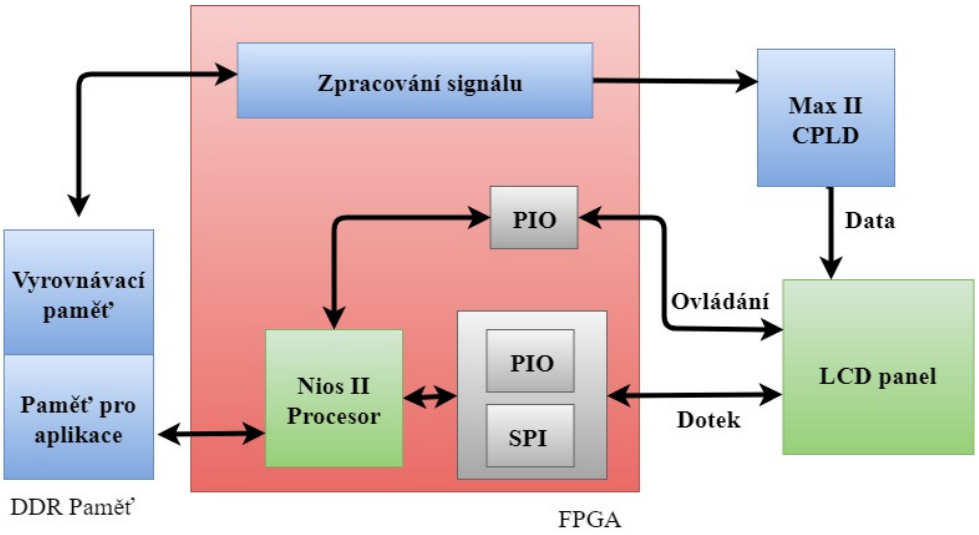
Softwarová část bude realizována v podprogramu Nois II SBT, v tomto prostředí bude probíhat následně i ladění úlohy.

6.2 Rozbor úlohy

Pro úspěšnou realizaci navržené úlohy je potřeba si ujasnit, jak by mělo zařízení pracovat, aby bylo možné efektivně navrhnout řídící systém.

6.2.1 Hardwarová struktura

Základním ovládacím prvkem této úlohy je samotný procesor Nios II, který zajišťuje jak ovládání LCD, tak i pamětí a dalšího procesu zpracování dat.



Obrázek 19: Blokové schéma LCD a jeho periferií [25]

DDR paměť o velikosti 256 MBitů je rozdělena na paměť programu a vyrovnávací obrázkovou paměť, ze které jsou data odesílána do bloku zpracovávajícího data pro zobrazení na LCD. K úpravě formátu dat pro jejich zobrazení na LCD dochází v bloku „zpracování signálu“. Data o jednotlivých pixelech jsou sem autonomně přiváděna z obrázkové vyrovnávací paměti po 64 bitové sběrnici s formátem 0:R:G:B:0:R:G:B. Průchodem dat tohoto bloku dochází ke změně formátu na 32 bitová data formátu 0:R:G:B, později je odebráno 8 bitů obsahujících znak „0“, který dle komunikačního protokolu symbolizuje začátek nového balíčku dat, v této chvíli jsou data 24 bitová. Z důvodu úspory pinů zařízení MAX II se data z bloku zpracovávajícího video signál odesílají po osmi bitech a původní 24 bitový formát je rozdělen na jednotlivé složky R, G, B, které jsou odesílány sériově za sebou vždy ve stejném pořadí.

Zařízení MAX II přijímá sériový tok dat a převádí je na zpátky na 24 bitový formát, který je následně odesílán do datového rozhraní pomocí konektoru propojujícího základní a multimediální desku. Logické obvody na základní desce FPGA jsou konstruovány na 2,5V, zatímco multimediální deska potřebuje pro správnou funkci 3,3V. Převod napěťových úrovní zajišťuje také zařízení MAX II.

LCD displej obsahuje 3 základní rozhraní:

- **Datové rozhraní**
  Zajišťuje příjem dat, která se v závislosti na ovládacích datech zobrazují na displeji. Data přicházejí z CPLD MAX II po 24 bitové sběrnici, ve formátu BGR (nejprve se přenáší modrá, poté zelená a nakonec červená složka signálu).

- **Ovládací rozhraní**
  Obsahuje řadič, který se stará o řízení a provoz LCD modulu. Data jsou přijímána z třívodičové sítě prostřednictvím PIO, které jsou připojeny k FPGA čipu.

- **Dotykové rozhraní**
  Pomocí SPI a PIO se ovládá periferie pro dotek. Modul s LCD obsahuje čip AD7843, který zpracovává data z digitizéru displeje a prostřednictvím SPI je posílá do hradlového pole. Protože jsou tato data přijímána i ve chvílích, kdy se uživatel displeje nedotýká, nejsou tedy vždy validní a není možné s nimi pracovat. Proto se zavádějí PIO, které indikují události typu „pen_up“ nebo „pen_down“ na jejichž základě je možné definovat přesnou chvíli, kdy jsou data validní a lze s nimi pracovat. [25]

6.3 HW komponenty demo úlohy

Nios II Processor – Symbolizuje samotný procesor, který je základní součástí každého návrhu. Nastavení procesoru umožňuje volbu jeho rychlost a velikost ze tří výrobcem stanovených možností.

- Nios II/e – Ekonomická konfigurace jádra, která zajistí nejmenší možné rozměry jádra, bohužel ale za cenu omezených funkcí a možností nastavení.

- Nios II/s – Standardní nastavení, které poskytuje malé rozměry jádra při zachování výkonu.

- Nios II/f – Při tomto nastavení je dosaženo nejvyššího výkonu a největšího počtu konfiguračních možností pro ladění funkcí procesoru.

V tomto projektu se využívá nastavení Nios II/f pro dosažení nejvyššího výkonu procesoru.

Spolu s adresou vektoru přerušení a výjimek je nutné nastavit také paměť, na kterou se bude systém v těchto případech odkazovat.

Další možnou konfigurací je nastavení pro ladění pomocí modulu JTAG. Systém umožňuje uživateli výběr ze čtyř úrovní. Čím vyšší úroveň je zvolena, tím více ladících prostředků je možno používat. Mnoho z nich je ale omezeno verzí používaného softwaru či koupí dalších rozšíření, proto je v tomto projektu využito první úrovně, která nabízí pouze připojení se pomocí JTAG, nahrání softwaru a použití softwarových breakpointů.

System ID Peripherial – Je nutné jej využívat při návrhu systému, který zahrnuje více procesorů, protože zabraňuje nesouladu mezi konfigurací hardwaru a softwaru. Při kompilaci softwaru program kontroluje, jestli se ID kompilovaného projektu shoduje s ID projektu navrženém pomocí Qsys.

Při běžném návrhu systému pomocí Qsys je přidání této komponenty vyžadováno, lze se tomu ale vyhnout úpravou nastavení projektu.

Generic Tri state Controller – umožňuje vytvořit ovladač pro čip, pro využívání dalších zařízení, která nejsou součásti samotného FPGA. Tento modul obsahuje parametry, které se nastavují podle potřeby a připojeného zařízení. Pokud se v návrhu vyskytuje více zařízení, se kterými je prováděna komunikace, čip rozhoduje pomocí pinu Tri-State Conduit.

Tri-State Conduit Pin Sharer – zajišťuje multiplexování signálů z již vytvořeného ovladače. Pro správné zpracování těchto signálů je při vkládání multiplexoru nutné zadat přesný počet signálů a k nim patřičný identifikátor.

Avalon ALTPLL – kontrolní blok, který vyrovnává fáze signálů clk. Jedná se o systém, který automaticky upravuje fázi lokálně generovaného signálu clk v závislosti na fázi vstupního clk signálu. V této komponentě je možné nastavit až 5 signálů clk o dané frekvenci, které mohou být generovány s posunem fáze dle nastavení uživatele.

PLL = Phase Locked-Loop, jedná se o velice krátkou smyčku, do které se systém dostane ve chvíli, kdy se vstupní signál neshoduje se signálem výstupním. V této smyčce se každá změna na vstupu projevuje jako změna fáze na výstupu. Používá se při návrhu vysokorychlostních aplikací.

Interval timer – časovač generující přerušení v požadovanou chvíli.

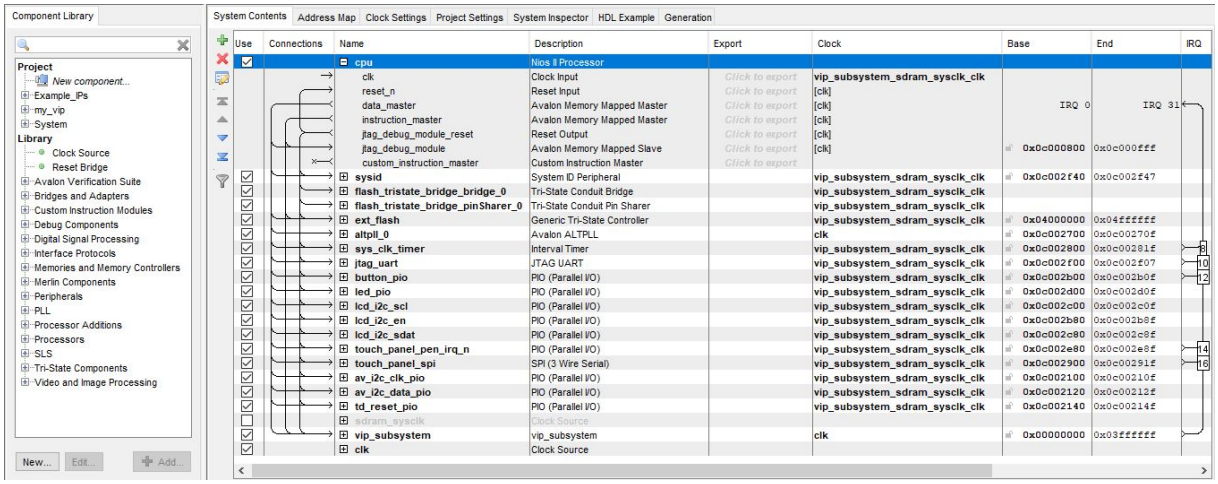
JTAG – rozhraní, které implementuje komunikaci mezi hostitelským počítačem a FPGA čipem. Komunikace je zajištěna prostřednictvím JTAG pinů na desce FPGA a přidružených obvodů. Pro detekci zařízení, správné kódování a dekódování tohoto signálu je zapotřebí využívat základní ovládací software firmy Altera, který je instalován do hostitelského počítače spolu s vývojovým prostředím Quartus.

Paralel I/O blocks – bloky, pomocí kterých je možné přidat spojení mezi procesorem Nios a periferními vstupně výstupními bloky. V nastavení těchto bloků je možné zvolit, kolik bitů budou mít jednotlivé bloky (např. daná úloha používá 4 LED, jsou potřeba 3 bity a poté je nutno přiřadit adresu každé LED, se kterými bude systém spojen). Tyto bloky jsou v tomto návrhu využívány pro ovládání programovatelných spínačů, komunikaci s LCD či dotykovou plochou a při využití sběrnice I^2C.

SPI (3 Wire Serial) – spolu s dalšími PIO bloky se stará o komunikaci zařízení AD7843, které vyhodnocuje, kdy se uživatel dotknul dotykové plochy LCD a zpracovává informace o každém z doteků, které jsou následně zpracovávány v FPGA.

Vip_subsystem – komponenta v níž dochází k vytváření přerušení dle požadavků systému a ke zpracování dat přicházejících po 32 bitové sběrnici z DDR paměti. Formát těchto dat je postupně upravován až k 8 bitové sběrnici, po které se odesílají data do zařízení MAX II.

Clock source – zdroj hodinového signálu.



Obrázek 20: Hardwarové součásti úlohy

Na obrázku 20 jsou viditelné použité komponenty a jejich vzájemné propojení. Na pravé straně ve sloupci IRQ můžeme vidět přiřazení priorit přerušení. Je možné si všimnout také sloupce, který udává hodinový signál, ke kterému jsou dané komponenty připojeny. Hodinový signál „vip_subsystem_sdram_sysclk_clk“ s frekvencí 83,3 MHz přichází do všech komponent kromě „vip_subsystem“ a „Avalon ALTPLL“.

Avalon ALTPLL využívá základního hodinového signálu clk o frekvenci 50 MHz a na jeho základě generuje další hodinové signály včetně „vip_subsystem_sdram_sysclk_clk“ dle uživatelského nastavení.

Do komponenty vip_subsystem přichází základní hodinový signál clk, ale také hodinový signál „vip_subsystem_sdram_sysclk_clk“, protože pro zapisování a čtení z paměti je použit právě tento hodinový signál. [26], [26], [27], [28]

7 Implementace demo úlohy do vývojového kitu Nios II

Implementace demo úlohy probíhá podobně jako tvorba vlastního projektu. Prvním krokem pro implementaci demo úlohy je otevření vývojového prostředí Quartus II a načtení projektu z adresáře v němž je uložena demo úloha. Po spuštění vývojového prostředí se zobrazí vyskakovací okno, ve které zvolíme možnost „Open Existing Project", vybereme adresář s uloženou demo úlohou a otevřeme soubor .qpf (Quartus Project File).

V této chvíli je vhodné provést kontrolu návrhu HW v programu Qsys a vygenerovat konfigurační soubor .qsys. Přesuneme se tedy do podprogramu Qsys, v záložce file zvolíme možnost open a vybereme soubor „cycloneIII_3c25_niosII_video_qsys.qsys". Zkontrolujeme propojení jednotlivých HW součástí, přiřazené adresy, frekvence hodinových signálů, priority přerušení jednotlivých komponent a můžeme se přesunout do záložky „Generate", kde vygenerujeme soubor .sof, který budeme nahrávat do připojeného zařízení.

Po vygenerování tohoto souboru se přesuneme zpátky do vývojového prostředí Quartus II a po vložení souborů dle postupu v sekci 5.2.1, můžeme pomocí podprogramu „Pin planner" zkontrolovat přiřazení jednotlivých pinů podle datasheetu. V tomto projektu jsou jednotlivé piny přiřazeny správně, proto můžeme projekt kompilovat a nahrávat do FPGA čipu.

Za pomocí záložky „Tools" se přepneme do podprogramu „Programmer", ve kterém vybereme námi vygenerovaný soubor s příponou .sof. Pomocí tlačítka „Hardware Setup" otevřeme okno, sloužící pro volbu připojeného zařízení a klikneme na tlačítko „Start". V této chvíli dojde k nahrání konfiguračního souboru do zařízení a můžeme se přesunout k samotnému oživení demo úlohy.

8 Oživení demo úlohy

Samotné oživení demo úlohy se provádí ve vývojovém prostředí Build Tools for Eclipse (SBT), které je součásti Quartus II. Při spuštění zvolíme adresář obsahující dosavadně vytvořené soubory a potvrdíme tlačítkem „Ok“. Protože demo úloha již obsahuje zdrojové kódy, vytvoříme prázdné projekty jako v předešlém návrhu a naimportujeme do nich soubory.

Proces zakládání projektů opět rozdělíme do 2 fází:

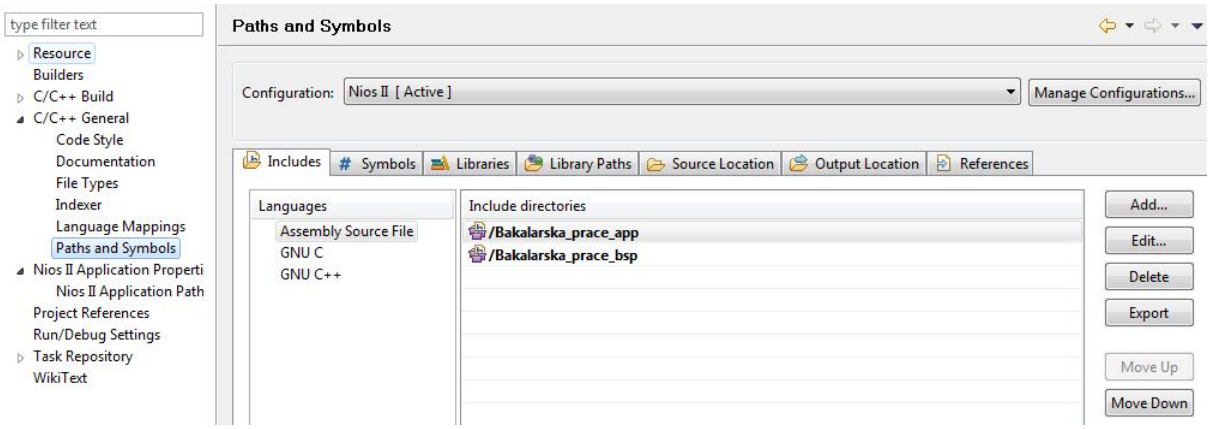
- **BSP projekt**
  Stejně jako při zakládání projektu v předešlé úloze zvolíme při vytváření projektu „**Nios II Board Support Package**“, opět zvolíme adresář, ve kterém máme uložen projekt a vybereme soubor s příponou .sopcinfo. Projektu přiřadíme vhodný název a klikneme na tlačítko „Finish“.

- **APP projekt**
  V záložce nový tentokrát zvolíme „Nios II Application“, protože nepotřebujeme vytvářet žádné výchozí soubory, navržené výrobcem. V okně, které se otevře po zvolení typu projektu, zvolíme název a přidáme koncovku _app. Před dokončením zakládání projektu pomocí tlačítka „Finish“ je potřeba zadat umístění BSP souboru, ten je jako v předešlé úloze uložen v projektu s příponou _bsp.

Po úspěšném založení projektů otevřeme v průzkumníku souborů složku s demo úlohou a ve složce „Software“ nalezneme dvě složky, které obsahují soubory pro jednotlivé projekty.

Import souborů je jednoduchý, stačí otevřít jednu ze složek, označit celý její obsah a přesunout do prostředí STB na příslušný projekt (obsah složky „LCD_default_app“ do projektu s příponou _app). To stejné je nutné provést i u druhého projektu.

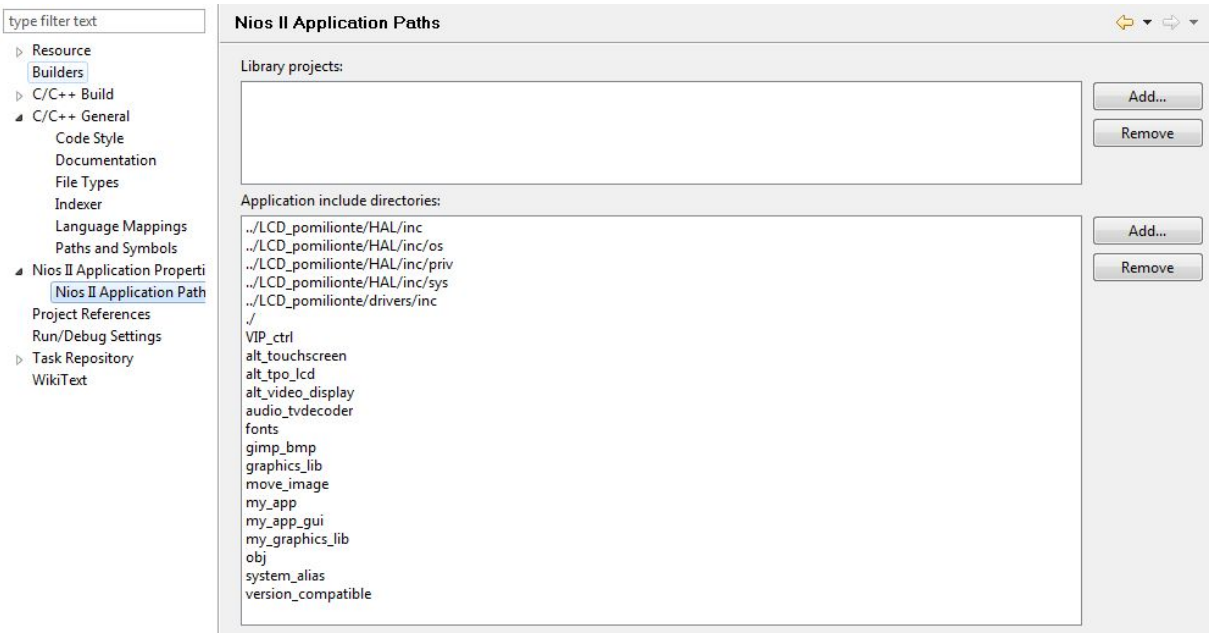
V předchozí úloze se v této fázi vygeneroval BSP soubor a pokračovalo se v nahrávání úlohy do připojeného zařízení.Jelikož je tento projekt rozdělen do několika souborů, je potřeba nastavit kompilátor tak, aby pracoval se všemi soubory. Kliknutím pravým tlačítkem myši na projekt s příponou _app se můžeme dostat do vlastností tohoto projektu, zvolíme tedy možnost „Properties“. Pro nastavení kompilátoru se přesuneme do záložky „C/C++ General“ a zvolíme položku „Paths and Symbols“



Obrázek 21: Nastavení kompilátorů

Na obrázku 21 vidíme vyskakovací okno, které se objeví po provedení dříve uvedeného postupu. Pro zajištění správné funkce je nutné nastavit jak kompilátor GNU C, tak i GNU C++, které jsou zobrazeny ve sloupečku s názvem „Languages“. Rozklikneme jeden z kompilátorů a pomocí tlačítka „Add“ se dostaneme k volbě adresářů s jednotlivými soubory. Pomocí tlačítka „Workspace“ se dostaneme do nabídky, ve které postupně přidáme oba námi vytvořené projekty. Stejný postup provedeme i u zbývajícího kompilátoru.

Jednotlivé zdrojové kódy jsou pro přehlednost uloženy ve složkách korespondujícími s názvy souborů. Podobně jako v předešlém kroku je nutné přiřadit konkrétní adresáře v záložce „Nios II Application Paths“, která nabízí přepnutí se do podokna „Nios II Application Properties“.



Obrázek 22: Vkládání zdrojových souborů

Na obrázku 22 je viditelné, že knihovny jsou vkládány po jedné. Tento postup, na rozdíl od vložení složky obsahující celý projekt, se mi při testování osvědčil a nedocházelo k problémům při kompilaci projektu.

Přidávání knihoven stačí provést pouze v kolonce „Library projects“ za pomocí tlačítka „Add“ a následného vybrání vhodné složky s knihovnou. Po zvolení adresáře budete dotázáni, jestli si přejete převést cestu na relativní cestu, tím je myšleno, jestli je potřeba ukládat kompletní cestu k adresáři (E/:Programování/ FPGA/ Bakalářská práce/ Projekt_r1/ alt_video_display) nebo relativní cestu k adresáři (.../ alt_video_display). V tomto případě stačí relativní adresa, proto potvrdíme tlačítkem „Yes“. Potvrdíme veškeré změny v nastavení a přistoupíme ke tvorbě BSP souboru.

Postup je stejný jako v projektu Hello from Nios II, klikneme tedy pravým tlačítkem myši na projekt s příponou _bsp a v záložce „Nios II“ zvolíme položku „Generate BSP“. Po vygenerování tohoto souboru klikneme opět pravým tlačítkem myši na projekt s příponou _app a v záložce „Run As“ zvolíme možnost „Nios II Hardware“, systém provede kontrolu návrhu, vytvoří makefile a začne programovat zařízení. Po dokončení tohoto procesu je projekt kompletně nahrán a oživen na připojeném zařízení.

8.1 Stav demo úlohy po spuštění

Dle popisu demo úlohy na stránkách by se po spuštění měl na displeji zobrazit rámeček s fotkou, se kterým je možné pohybovat a měnit jeho velikost. V záhlaví by měl být zobrazen čas, který je možné nastavit pomocí programovatelných tlačítek.

Obrázek 23: Srovnání skutečného výsledku s výsledkem na webových stránkách [29]

Demo úloha měla po nahrání do zařízení vypadat jako na obrázku vlevo, ve skutečnosti ale vypadala jako na obrázku vpravo. Jak je vidět na fotce, žádný obrázek v rámečku se nezobrazil, dotek nefungoval, nezobrazoval se čas ani text podle popisu na webových stránkách. [29]

8.2 Analýza demo úlohy

Vzhledem k tomu, že k demo úloze nebyla poskytnuta žádná dokumentace s popisem jednotlivých funkcí, bylo nutné ladit program pomocí debug režimu. Po spuštění programu v tomto režimu, bylo zjištěno, že se program během zpracovávání jednoduchých instrukcí zacyklí a nepokračuje dál, proto se na displeji nezobrazila požadovaná data a zařízení nereagovalo na dotyk.

Pro výpis aktuálního času a jednotlivých znaků byla využívána funkce „update_graphics“ s parametrem „write_all“, na kterém záviselo, jestli se při průběhu programu funkcí budou vypisovat jednotlivé znaky a řetězce nebo se bude aktualizovat čas. Tyto operace byly prováděny za pomocí funkce „snprintf“, ve které se program zacyklil při pokusu o alokaci paměti pro daný řetězec či znak. Kvůli tomuto problému program nepokračoval.

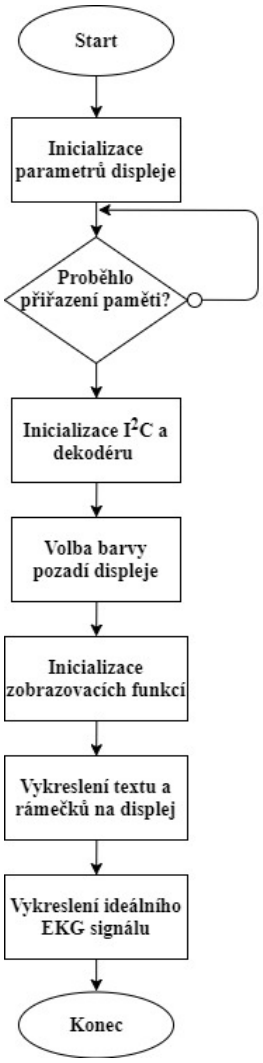
Po opětovném spuštění programu a odstranění veškerých výskytů dříve zmíněné funkce, bylo na displeji viditelné stejné zobrazení jako před úpravou zdrojového kódu, z čehož plyne, že jediným výsledkem demo úlohy je inicializace displeje a zobrazení dvou rámečků.

9 Tvorba demonstrační úlohy

Pro vytvoření demonstrační úlohy podle návrhu s použitím demo úlohy bylo nutné dále analyzovat zdrojové kódy a testovat ostatní funkce. Vzhledem k tomu, že se na displeji zobrazily dva rámečky, bylo vhodné zaměřit se na část kódu, která zpracovává jejich vykreslování. Jedním ze zdrojových souborů je „simple_graphics“, který obsahuje funkce, pro vykreslování obrazců na displej.

9.1 Funkce využité v návrhu demonstrační úlohy

Pro zobrazení požadovaných obrazců je nutné systematicky nastavit barvu jednotlivých pixelů tak, abychom dostali výsledek podle představ. Jelikož displej obsahuje 384 000 pixelů, je vhodné zjednodušit si ovládání jednotlivých pixelů pomocí dalších funkcí, které jsou navrženy tak, aby na základě zadaných parametrů vytvořily dané obrazce.



Obrázek 24: Vývojový diagram demonstrační úlohy

9.1.1 Inicializační funkce

Před samotným použitím LCD je potřeba provést inicializaci a nastavit parametry, tohle je zajištěno pomocí inicializačních funkcí.

alt_video_display* alt_video_display_only_frame_init (int width, int height, int color_depth, int num_buffers); - Inicializuje parametry displeje.

void init_i2c (void); - Nastaví parametry pro komunikaci prostřednictvím I^2C.

void tv_decoder_init (void); - Kontroluje nastavené parametry přenosu pomocí I^2C.

int AsInitTpoLcd(alt_tpo_lcd* lcd_serial) - Ukládá adresy do strukturovaných proměnných pro zajištění komunikace displeje s čipem. Do parametru se uvádí struktura, která obsahuje požadované proměnné.

void set_frame_color(alt_video_display* display, int col); - Vykreslí pozadí displeje ve zvolené barvě. Parametr col udává, jakou barvu bude pozadí mít. Do prvního parametru se uvádí proměnná, ve které je uložena návratová hodnota první inicializační funkce.

void Frame_Reader_set_frame_0_properties (int base_address, int words, int samples, int width, int height, int interlaced); - Funkce inicializující hladinu pro vykreslování na displeji. Do proměnné „words“ se vkládá počet pixelů, které budou ovládány v dané hladině, toto číslo se používá také pro počet vzorků v proměnné „samples“. Poté je nutné do proměnných „width“ a „height“ uvést celkovou výšku a šířku displeje.

void Frame_Reader_switch_to_pb0(void); - Přepíná hladiny, ve kterých se na displeji zobrazují data.

void Mixer_set_layer_position (int layer, int x, int y); - Nastaví kurzor do alfanumericky uvedené hladiny v prvním parametru na konkrétních souřadnicích.

Ostatní inicializační funkce slouží zavedení výchozích parametrů funkcím ohraničujícím prostor na LCD, který je možné v danou chvíli ovládat.

9.1.2 Ovládací funkce

void vid_set_pixel(int horiz, int vert, int color, alt_video_display* display); - Slouží k ovládání jednotlivých pixelů displeje. Do parametrů uvádíme horizontální a vertikální souřadnice konkrétního pixelu, jeho barva a struktura, ve které jsou definovány parametry displeje.

int vid_draw_box (int horiz_start, int vert_start, int horiz_end, int vert_end, int color, int fill, alt_video_display* display); - Vykreslí obdélník na zadaných souřadnicích. Je potřeba zadat počáteční a koncové souřadnice na vertikální a horizontální ose, parametrem „fill" se volí, zda bude obdélník vybarven celý nebo se vykreslí pouze jeho obrys (při zadání čísla 0 bude vykreslen pouze obrys, zatímco číslo 1 zajistí vytvoření obdélníku s výplní dané barvy). Dále je nutné zvolit barvu obrazce a opět strukturu s parametry displeje.

int vid_print_string_alpha(int horiz_offset, int vert_offset, int color, char *font, alt_video_display* display, char string[]); - S použitím zdrojového souboru, ve kterém je definován konkrétní zvolený font vypíše řetězec znaků na uživatelem zadané souřadnice.

9.2 Sestrojení demonstrační úlohy

Po analýze veškerých zdrojových souborů a testování jednotlivých funkcí bylo zobrazení podle požadavků uvedených v návrhu úlohy provedeno za použití ovládacích funkcí. Pro zajištění přehlednosti hlavní funkce main(), byly vytvořeny 2 funkce, kterými je dosaženo požadovaného zobrazení.

9.2.1 void EKG_init(int H, int V);

V této funkci se vyskytují 2 parametry, určující pozici vykreslování demonstračního EKG signálu.

- H - určuje posun na horizontální ose.
- V - určuje posun na vertikální ose.

Výchozí pozice pro vykreslování je nastavena na 0 x 240 pixelů, což znamená, že se signál začíná vykreslovat od nultého pixelu na horizontální ose a 240 pixelu na ose vertikální.

Funkce vid_set_pixel se při testování zdála být nefunkční, ale po čase bylo zjištěno, že změna barvy jednoho pixelu není viditelná lidským okem ani při pohledu zblízka. Proto je tato funkce zakomponovaná do cyklů for.

```
for (h = 0; h <= 5; h++){
    for (v = 0; v <= 3; v++){
        vid_set_pixel(40 + h + H, 238 + v + V, RED_24, display);
    }
}
```

Obrázek 25: Část funkce pro vykreslování demonstračního EKG signálu

Prvním cyklem s použitou proměnnou „h" je zajištěno ovládání většího počtu pixelů na horizontální ose. Tento parametr prakticky udává, jak širokou čarou bude demonstrační EKG signál vykreslován. Pro tuto úlohu se hodnota 5 ukázala jako dostačující.

V druhém cyklu je použita proměnná „v", která má stejné využití jako proměnná v předešlém

cyklu, ale na ose vertikální. Hodnota této proměnné se v průběhu vykreslování signálu mění podle potřeby.

V samotné funkci jsou poté k pevně stanoveným souřadnicím přičítány hodnoty z proměnných, které jsou obsaženy v cyklech a ve vstupních parametrech funkce.

9.2.2 void Frame_init(void);

Vykreslí veškeré rámečky a řetězce znaků přesně podle zadání.

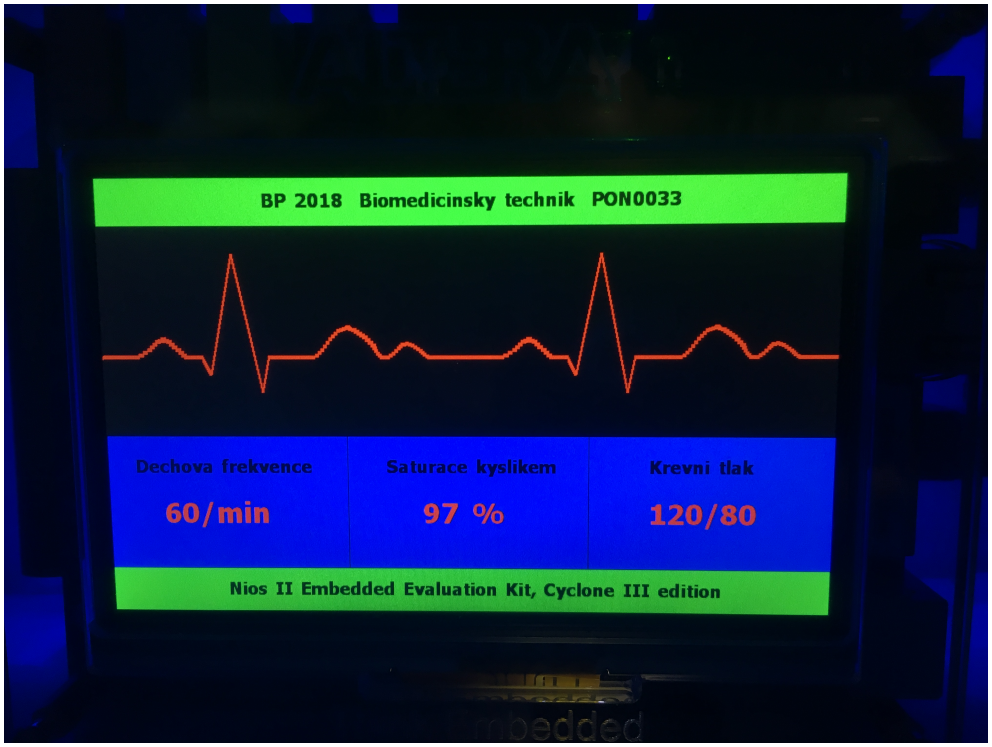
```
// Spodní rámeček s popisem
vid_draw_box (0, 430, 800, 480, BLUE_16, 1, display);
vid_print_string_alpha(130, 440, 0x00, BLUE_16, tahomabold_20, display,
        "Nios II Embedded Evaluation Kit, Cyclone III edition");

// Horní rámeček s popisem
vid_draw_box (0, 0, 800, 50, BLUE_16, 1, display);
vid_print_string_alpha(180, 10, 0x00, BLUE_16, tahomabold_20, display,
        "BP 2018  Biomedicinsky technik   PON0033");

// Dechová frekvence
vid_draw_box(0, 280, 266, 429, BLUE_24, 1, display);
vid_print_string_alpha(30, 300, 0x00, BLUE_24,tahomabold_20,display,
        "Dechova frekvence");
vid_print_string_alpha(60, 345, RED_24, BLUE_24,tahomabold_32,display,
            "60/min");

// SpO2
vid_draw_box(267, 280, 533, 429, BLUE_24, 1, display);
vid_print_string_alpha(310, 300, 0x00, BLUE_24, tahomabold_20, display,
        "Saturace kyslikem");
vid_print_string_alpha(350, 345, RED_24, BLUE_24,tahomabold_32,display,
                "97 %");

// Krevní tlak
vid_draw_box(534, 280, 800, 429, BLUE_24, 1, display);
vid_print_string_alpha(600, 300, 0x00, BLUE_24, tahomabold_20, display,
        "Krevni tlak");
vid_print_string_alpha(600, 345, RED_24, BLUE_24,tahomabold_32,display,
                "120/80");
```

Obrázek 26: Tělo funkce pro zobrazení rámečků a řetězce znaků

Rámečky jsou vykreslovány pomocí funkce vid_draw_box na zadaných souřadnicích v barvách jejichž hexadecimální hodnoty jsou uloženy v makrech.

Použitím funkce vid_print_string_aplha je zajištěn výpis znakových řetězců. Znaky jsou vypisovány fontem tahomabold o velikost 20 a 32 bodů.

Posledním krokem je opětovná implementace a oživení úlohy podle postupu zmíněném v sekcích 7 a 8.



Obrázek 27: Grafický výstup demonstrační úlohy

10 Zhodnocení dosažených výsledků

Prvotním cílem této bakalářské práce bylo zpracování rešerše na téma srovnání platforem výrobců Altera a Xilinx. Informace byly čerpány z webových zdrojů a konferenčních článků, protože věděcké články se zabývají spíše použitím určitého FPGA v dané úloze a nezmiňují možnost použití jiného vývojového kitu či FPGA čipu pro řešení stejné úlohy. Při srovnání architektur Virtex-5 a Stratix III bylo zjištěno, že Xilinx používá LUT se šesti vstupy, zatímco ALM v zařízení od výrobce Altera poskytuje 8 sdílených vstupů, což staví Alteru do nevýhodné pozice, protože při implementaci jednodušší funkce, která nevyužívá všechny vstupy, dochází ke ztrátě nepoužitých vstupů, jelikož je na rozdíl od Xilinxu nelze používat odděleně. Naproti tomu při testování výkonu těchto platforem se Altera ukázala jako výkonnější.

Další část práce se zabývá využitím vývojového kitu v biomedicíně. Díky relativně malým rozměrům a rychlým inovacím těchto čipů bylo navrženo použití kitu pro monitoring životních funkcí, zajišťující náhradu návštěv lékaře v případě, kdy nejsou nutné, čímž dojde k zefektivní zdravotnického systému v zemích, kde je nedostatek lékařů. Tento monitoring může omezit návštěvnost lékařů zdravými pacienty a zajistit tak více prostoru pro vyšetření a stanovení dalšího postupu léčby u nemocných pacientů. Dalším zmíněným využitím je implementace umělé neuronové sítě, která má schopnost učit se a reagovat na různé odchylky při snímání EEG či EKG signálu. Při vhodném použití se očekává snížení chyby, způsobené lidským faktorem při vyhodnocování odchylek signálů.

Návrh demonstrační úlohy byl časově nejnáročnější, protože podobný, kompletně funkční návrh nebyl doposud zpracován a zveřejněn, proto i malý problém pozastavil průběh realizace na velmi dlouhou dobu. Zásadním problémem byla instalace ovladače pro komunikaci PC s vývojovým kitem, řešení tohoto problému byla většinou krátkodobá z důvodu užívání Windows 10, který není touto verzí softwaru podporován, proto bylo potřeba na stávajícím zařízení vytvořit virtuální počítač s Windows 7, na kterém už všechno fungovalo, jak má. Problém nastal při oživování demo úlohy, konkrétně v nastavení kompilátorů a adresářů se zdrojovými kódy tak, aby systém fungoval. Po oživení demo úlohy byla zjištěno, že f, jak bylo funguje v rozporu s informacemi uvedenými tvůrcem. Vzhledem k tomu, že k úloze nebyla poskytnuta žádná dokumentace, ve které by byly popsány jednotlivé principy, bylo nutné analyzovat zdrojové kódy a na tomto základě začít tvořit demonstrační úlohu.

Grafický výstup praktické části této bakalářské práce je možno vidět na obrázku 27. Úloha byla vypracována přesně podle zadaného návrhu a při její implementaci bylo využito 24 603 logických buněk z celkového počtu 24 624. Z toho vyplývá, že návrh využívá téměř veškerou logickou kapacitu vývojového kitu. Fyzických pinů FPGA bylo využito 193 z 216, zbylých 19 volných pinů by mohlo být použito například jako vstupy pro další zpracování externích signálů.

11 Závěr

Smyslem této bakalářské práce bylo začlenění vývojového kitu Nios II do oboru biomedicínského inženýrství, seznámení čtenářů s problematikou hradlových polí a tvorbou návodu v podobě demonstrační úlohy, na jejímž základě budou čtenáři schopni modifikovat dosavadní návrh, či implementovat vlastní úlohu do vývojového kitu Nios II.

Teoretická část obsahuje popis základní struktury FPGA čipů a přehled nejčastěji používaných programovacích jazyků, sloužících pro popis hardwaru. V této části je také možné najít konkrétní příklady využití vývojového kitu v biomedicíně.

Praktická část detailně popisuje návrh a realizaci jak základního projektu, který slouží jako učební příklad před samotnou realizací složitějších úloh, tak i demonstrační úlohy. Výstupem demonstrační úlohy je grafické rozhraní, které má vyhrazený prostor pro vykreslování EKG signálu a číselný výpis pro hodnoty tlaku, obsahu kyslíku v krvi a dechové frekvence.

Vzhledem k tomu, že vývojový kit obsahuje LCD s dotykovou plochou, v budoucnu je možné na tuto úlohu navázat a obohatit ji o funkce závislé na vyhodnocení dotyku. Součástí projektu přiloženém na CD jsou zdrojové soubory potřebné pro ovládání dotykové plochy, které je možno využít. Díky nevyužitým pinům v návrhu úlohy je možné připojit k zařízení externí signály například z vyšetřovacích systémů a navrhnout algoritmus pro jejich zpracování. Těmito vylepšeními je možné z demonstrační úlohy, se zobrazenými ideálními daty, vytvořit reálný monitor životních funkcí.

12 Použitá literatura

[1] *DANĚK, Martin. Programovatelná hradlová pole – FPGA. Automa – časopis pro automatizační techniku, s. r. o.* [online]. Děčín: Ústav teorie informace a automatizace Akademie věd ČR, 2006, 2006(2) [cit. 2017-10-19]. Dostupné z: http://automa.cz/cz/casopis-clanky/programovatelna-hradlova-pole-fpga-2006_02_30930_672/

[2] *PECH, Jan. Nebojte se FPGA. HW server s.r.o.* [online]. HW server, 2002 [cit. 2017-10-19]. Dostupné z: https://vyvoj.hw.cz//teorie-a-praxe/dokumentace/nebojte-se-fpga.html

[3] *2.3 Obvody FPGA. Ústav radioelektroniky* [online]. [cit. 2017-10-19]. Dostupné z: http://www.urel.feec.vutbr.cz/ kolouch/pld/1_prednasky/kapitola02_03.html

[4] *PECH, Jan. Programovatelné logické obvody – část 1. DPS Elektronika od A do Z* [online]. Liberec: CLIQUO & Binteractive, 2014, 2014(3) [cit. 2017-10-19]. ISSN 1805-5044. Dostupné z: http://www.dps-az.cz/vyvoj/id:2404/programovatelne-logicke-obvody-cast-1

[5] *PECH, Jan. Programovatelná logika – část 5: Jazyky pro popis logických obvodů. DPS Elektronika od A do Z* [online]. CLIQUO & Binteractive, 2015, 2015(1) [cit. 2017-10-19]. ISSN 1805-5044. Dostupné z: http://www.dps-az.cz/vyvoj/id:22197/programovatelna-logika-cast-5-jazyky-pro-popis-logickych-obvodu

[6] *PECH, Jan. Programovatelná logika a HDL – část 3: Základy vývoje pro FPGA. DPS Elektronika od A do Z* [online]. CLIQUO & Binteractive, 2014, 2014(5) [cit. 2017-10-19]. ISSN 1805-5044. Dostupné z: http://www.dps-az.cz/vyvoj/id:12031/programovatelna-logika-a-hdl-cast-3-zaklady-vyvoje-pro-fpga

[7] *ACH, TA PAMĚŤ. . . . Učíme se VHDL* [online]. 2015 [cit. 2017-10-19]. Dostupné z: https://vhdl.cz/ach-ta-pamet/

[8] *KOLOUCH, Jaromír. Možnosti realizace pamětí RAM v obvodech FPGA. Elektrorevue* [online]. Brno: International Society for Science and Engineering, o.s., 2003, 2003(27) [cit. 2017-10-19]. ISSN 1213-1539. Dostupné z: http://www.elektrorevue.cz/clanky/03027/index.html

[9] *KAŠÍK, Vladimír. Prostředky pro navrhování logických systémů pro obvody FPGA. Automa – časopis pro automatizační techniku, s. r. o.* [online]. Děčín, 2006, 2006(7) [cit. 2017-10-19]. Dostupné z: http://automa.cz/cz/casopis-clanky/prostredky-pro-navrhovani-logickych-systemu-pro-obvody-fpga-2006_07_31260_2910/

[10] *Design a Sequential Multiplier* [online]. 2012 [cit. 2017-10-19]. Dostupné z: http://my-fpga.blogspot.cz

[11] *Xilinx schematic editor. Vesselyn* [online]. Papuk Group, 2017 [cit. 2017-10-19]. Dostupné z: http://vesselyn.com/eGlsaW54LXNjaGVtYXRpYy1lZGl0b3I/

[12] *ModelSim PE Evaluation Software. In: Mentor* [online]. [cit. 2018-04-29]. Dostupné z: https://mgc-images.imgix.net/fpga/modelsim_pe-75279295-fcb3-47b4-ab12-7b129bc4cb39.jpg

[13] *Xilinx vs. Altera: Calling the Action in the Greatest Semiconductor Rivalry. Electronic Engineering Journal* [online]. 2014 [cit. 2017-10-19]. Dostupné z: https://www.eejournal.com/article/20140225-rivalry/

[14] *PERCEY, Andrew. Advantages of the Virtex-5 FPGA 6-Input LUT Architecture. In: Xilinx* [online]. 2007, 19. December [cit. 2018-01-03]. Dostupné z: http://notes-application.abcelectronique.com/077/77-43425.pdf

[15] *LUJAN, C. A., F. J. MORA a J. D. MARTINEZ. Comparative analysis between the Stratix II (Altera) and Virtex 4 (Xilinx) for implementing a LVDS bus receiver. 2007 4th International Conference on Electrical and Electronics Engineering* [online]. IEEE, 2007, , 373-376 [cit. 2017-10-19]. DOI: 10.1109/ICEEE.2007.4345043. ISBN 978-1-4244-1165-8. Dostupné z: http://ieeexplore.ieee.org/document/4345043/

[16] *MARKOV, Igor. What are the major differences between Xilinx and Altera FPGAs, e.g., do they compete in the exact same application cases, is one more competitively priced, is either more user friend with its license less expensive to license, etc., dev. env.? Quora* [online]. 2014 [cit. 2017-10-19]. Dostupné z: https://www.quora.com/What-are-the-major-differences-between-Xilinx-and-Altera-FPGAs-e-g-do-they-compete-in-the-exact-same-application-cases- is-one-more-competitively-priced-is-either-more-user-friend-with-its-license-less-expensive-to-license-etc-dev-env#

[17] *Xilinx vs Altera Update 2017. SemiWiki* [online]. 2017 [cit. 2017-10-11]. Dostupné z: https://www.semiwiki.com/forum/content/6602-xilinx-vs-altera-update-2017-a.html

[18] *Stratix III FPGAs vs. Xilinx Virtex-5 Devices. In: Intel* [online]. San Jose: Altera Corporation, 2007 [cit. 2018-01-03]. Dostupné z: https://www.altera.com/en_US/pdfs/literature/wp/wp-01007.pdf

[19] *USB-Blaster*: Cable programming USB. *RS Components* [online]. United Kingdom: RS Components [cit. 2018-01-02]. Dostupné z: https://media.rs-online.com/t_large/F6904093-01.jpg

[20] *Multimedia-HSMC Card(Phased Out). Terasic* [online]. Taiwan [cit. 2018-04-25]. Dostupné z: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=240

[21] *NEEK Overview. In: Www.altera.com* [online]. USA: Altera Corporation, 2008 [cit. 2018-04-25]. Dostupné z: https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-cyc3-embedded.html

[22] *ROY CHOWDHURY, Shubhajit. FPGA-Based Clinical Diagnostic System using Pipelined Architectures in the Nios II Soft-Core Processor. Altera* [online]. Jadavpur University, Calcutta: Altera Corporation, 2007 [cit. 2018-04-27]. Dostupné z: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/dc/_2007/i3.pdf

[23] *JUSTIN, Čermák. Implementace umělé neuronové sítě do obvodu FPGA.* Brno, 2011. Diplomová práce. Vysoké učení technické. Vedoucí práce Ing. Marek Bohrn.

[24] *Nios II System Architect Design Tutorial. Altera* [online]. San Jose: Altera Corporation, 2011, 6.2011 [cit. 2018-04-25]. Dostupné z: https://www.altera.com/en_US/pdfs/literature/tt/tt_nios2_system_architect.pdf

[25] *Implementing an LCD Controller. Altera* [online]. San Jose: Altera Corporation, 2008, 5.2008 [cit. 2018-04-25]. Dostupné z: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an527.pdf

[26] *Avalon Tri-State Conduit Components User Guide. Altera* [online]. San Jose: Altera Corporation, 2011, 5.2011 [cit. 2018-04-25]. Dostupné z: https://www.altera.com/en_US/pdfs/literature/ug/ug_avalon_tc.pdf

[27] *ALTPLL (Phase-Locked Loop) IP Core User Guide. Altera* [online]. San Jose: Altera Corporation, 2017, 16. 6. 2017 [cit. 2018-04-25]. Dostupné z: https://www.altera.com/en_US/pdfs/literature/ug/ug_altpll.pdf

[28] *Nios II Classic Processor Reference Guide. Altera* [online]. San Jose: Altera Corporation, 2016 [cit. 2018-04-25]. Dostupné z: https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii5v1.pdf

[29] *Altera's Video and Image Processing Suite Demo on the NEEK. In: AlteraWiki* [online]. Altera Corporation, 2015, 6.12.2015 [cit. 2018-04-25]. Dostupné z: http://www.alterawiki.com/wiki/Altera's_Video_and_Image_Processing_Suite_Demo_on_the_NEEK

Seznam příloh na CD

I Bakalářská práce PON0033.pdf.....PDF soubor s dokumentací

II Bakalářská práce PON0033.rar.....Zdrojové soubory pro demonstrační úlohu